Titre: Title:	Algorithme de recherche directe pour l'optimisation robuste de fonctions bruitées
Auteur: Author:	Amina Ihaddadene
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ihaddadene, A. (2014). Algorithme de recherche directe pour l'optimisation robuste de fonctions bruitées [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1635/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1635/
Directeurs de recherche: Advisors:	Charles Audet, & Sébastien Le Digabel
Programme: Program:	Mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

ALGORITHME DE RECHERCHE DIRECTE POUR L'OPTIMISATION ROBUSTE DE FONCTIONS BRUITÉES

AMINA IHADDADENE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ALGORITHME DE RECHERCHE DIRECTE POUR L'OPTIMISATION ROBUSTE DE FONCTIONS BRUITÉES

présenté par : IHADDADENE Amina
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ADJENGUE Luc-Désiré, Ph. D., président
M. AUDET Charles, Ph. D., membre et directeur de recherche
M. LE DIGABEL Sébastien, Ph. D., membre et codirecteur de recherche
M. DELAGE Erick, Ph. D., membre

REMERCIEMENTS

Je souhaite naturellement adresser mes remerciements à mes directeurs de recherche, Charles Audet et Sébastien Le Digabel, pour leur encadrement de qualité, leur disponibilité et leur efficacité tout au long de mon travail, je vous en suis très reconnaissante. Un merci particulier à Charles pour son aide et pour sa bonne humeur constante. Merci à Sébastien pour ses commentaires judicieux pour la correction de ce mémoire, sans oublier l'aide informatique précieuse de Christophe Tribes .

Je désire remercier les membres du personnel du GERAD pour m'avoir fourni un cadre de travail agréable, plus particulièrement, merci à Pierre Girard et Edoh Liagros Logo pour leur soutien technique.

Je voudrais remercier sincèrement mes directeurs Charles et Sébastien pour leur aide financière.

Je remercie également Luc-Désiré Adjengue et Erick Delage d'avoir accepté d'évaluer ce travail, et pour m'avoir fait l'honneur de faire partie du jury.

Un grand merci à Sara Séguin, c'est un plaisir de t'avoir connue, ainsi que ton entourage, je te remercie pour toutes ces soirées culinaires chez toi. Merci à Mathilde pour ses conseils et pour les discussions enrichissantes durant ma maîtrise. Je remercie également mes collègues du bureau Mathieu, Nadir, Stéphane, Nicolas et Thibault pour la bonne ambiance qui régnait au bureau.

Le meilleur pour la fin ! Je désire remercier Imène Chahira leti Allab de m'avoir supporté toutes ces années, et pour son support moral. Elle n'a jamais cessé de m'encourager je te remercie pour tes précieux conseils. Je suis très reconnaissante envers Hocine Bouarab pour ses conseils de rédaction et son aide si précieuse durant mon cursus universitaire de Bbz à Poly. Enfin, je voudrais témoigner ma gratitude envers mes parents pour m'avoir encouragé sans cesse. Merci aussi aux autres membres de ma famille.

RÉSUMÉ

Ce mémoire considère les problèmes d'optimisation de type boîte-noire, pour lesquels les fonctions définissant le problème sont complexes, difficiles à évaluer ou leur expression analytique est inconnue. Pour la plupart des problèmes industriels, il s'agit d'un programme informatique complexe et souvent inaccessible pour des raisons de confidentialité. L'exploitation de ces fonctions se fait par l'analyse des réponses à des entrées données, elles sont considérées comme des *boîtes-noires* sans aucune hypothèse sur les propriétés des fonctions.

En optimisation de boîtes-noires, les méthodes mathématiques classiques ne sont pas applicables étant donné qu'elles reposent sur la différentiabilité, continuité, etc. C'est pourquoi plusieurs algorithmes sans dérivées conçus pour résoudre ce type de problèmes ont été développés et ne requièrent aucune hypothèse sur la fonction. La modélisation par les boîtes-noires ainsi que l'application des algorithmes sans dérivées ont permis de résoudre des problèmes d'optimisation des plus difficiles. Cependant, l'utilisation de ces algorithmes peut retourner des solutions affectées par le bruit.

Les problèmes considérés ici sont ceux où le bruit est présent autant sur les entrées que sur la sortie de la boîte-noire. La solution obtenue par des algorithmes d'optimisation n'est pas désirable. En effet, lorsque les solutions peuvent varier dans un intervalle comme la tolérance de production, une légère modification de la solution optimale dans son intervalle de tolérance peut détériorer la performance de cette solution. Dans le contexte où la fonction objectif est à minimiser, la solution donnant une petite valeur de l'objectif n'est pas celle qu'on recherche, on s'intéresse plutôt à une solution robuste. La robustesse ici se traduit par une petite variation de la valeur de l'objectif dans un voisinage de la solution proposée.

La contribution de ce mémoire est le développement d'un algorithme d'optimisation robuste, nommé RobustMads basé sur un algorithme de recherche directe, traitant les fonctions bruitées et retournant une solution robuste. Cet algorithme propose une façon judicieuse de lisser les valeurs de la fonction objectif pour corriger les valeurs bruitées. Dans le contexte où l'évaluation des fonctions est coûteuse, cette technique de lissage ne nécessite aucune évaluation supplémentaire.

Une solution robuste est celle ayant un large voisinage dans lequel la valeur de la fonction varie peu. On introduit quatre mesures différentes pour mesurer la robustesse d'une solution. L'analyse des résultats de l'application de RobustMads sur une batterie de fonctions tests, dont deux types de bruits, ont été testés, a montré que l'algorithme est capable de retourner des solutions robustes. De plus, plus le niveau de bruit est élevé, plus RobustMads améliore la solution.

ABSTRACT

This work considers blackbox optimization problems for which the objective function is complex, has no known analytical expression or may be costly to evaluate. In most industrial problems, the function may come from a complex software from which the code cannot be accessed for confidentiality purposes. The exploitation of such functions is done by analyzing its response to given inputs. They are considered blackbox functions on which no hypothesis is made about their properties. In blackbox optimization, classical mathematical methods cannot be applied because they rely on differentiability and continuity, just to name a few. For this reason, derivative-free optimization algorithms that do not require any assumption about the objective function are developed. Derivative-free algorithms and blackbox modeling have made it possible to solve some very hard optimization problems. Unfortunately, these algorithms may return a solution affected by the noise.

These are problems that have a stochastic noise presence on the inputs as well as the outputs. The optimal solution obtained from classical algorithms may not be desirable. For example, when there exist a known uncertainty on some input variables, a slight variation of the solution may deteriorate the solution. In the case of a minimization problem, one would be looking not only to minimize the function, but also to obtain a robust solution. Here, a robust solution is one for which the objective function undergoes a slight deterioration within its neighborhood. The main contribution of this work is the development of a robust optimization algorithm named RobustMads, based on a direct search algorithm and designed for functions that include a stochastic noise and that returns a robust solution. In a context where evaluating the objective function may be costly, this algorithm proposes a judicious approach to smoothing the values of the objective function that does not require any additional evaluation. A robust solution possesses a large neighborhood for which the objective function varies slightly. Four means of measuring the robustness of a solution are introduced. The analysis of the results of the application of RobustMads on a series of test functions, including two different types of noise, shows that the algorithm is capable of finding robust solutions. Our numerical experiments suggest that the higher the noise, the more RobustMads improves the solution.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CS	Coordinate Search.
GPS	Generalized Pattern Search .
MADS	Mesh Adaptive Direct Search (algorithme) .
VNS	Variable Neighbourhood Search.
LTMADS	Lower Triangular MADS (instance originale de MADS).
ORTHOMADS	MADS déterministe avec directions orthogonales (seconde instance de MADS).
NOMAD	Nonlinear Optimization by Mesh Adaptive Direct Search (logiciel).
LH	Hypercubes Latins .
NEWUOA	NEW Unconstrained Optimization Algorithm.

f	La fonction objectif.
z	La fonction lisse.
$\widetilde{f}$	La fonction lissée.
k	Compteur d'itérations.
x_k	Centre de sonde à l'itération k.
D_k	Ensemble des directions de sonde à l'itération k.
M_k	Ensemble des points du treillis à l'itération k.
P_k	Ensemble des points de sonde à l'itération k.
Δ_k	Taille de treillis à l'itération k pour GPS.
Δ_k^m	Taille de treillis à l'itération k.
Δ_k^p	Taille de sonde à l'itération k.
V_k	La cache à l'itération k.
W_k	Bloc des points générés à l'itération k.
I_k	Cardinalité de W_k dénotée $\mid W_k \mid$.

$\|x\|$	Norme euclidienne de x définie par $< x, x >^{1/2}$.
$\|x\|_\infty$	Norme infinie de x.
$\mathbb{E}$	Espérance mathématique de x.

CHAPITRE 1

INTRODUCTION

La plupart des problèmes d'optimisation, modélisant un cas réel dans divers domaines et notamment en ingénierie, sont définis par des fonctions difficiles à évaluer. Cette difficulté peut être associée au fait que : (1) le nombre de variables des fonctions est élevé comme des centaines ou voire des millions, (2) l'expression analytique des fonctions n'est pas connue car elle est décrite par un code informatique complexe incluant des simulations coûteuses, (3) l'expression analytique des fonctions est complexe. C'est sur les deux cas (2) et (3) qu'on se concentre dans le cadre de ce travail.

Lorsque les fonctions ont une expression complexe, les dérivées sont parfois difficiles à calculer et supposées inexistantes lorsqu'il est trop coûteux de les estimer. D'un point de vue mathématique, la résolution de ce type de problèmes est très compliquée et l'utilisation des méthodes classiques d'optimisation n'est pas possible car elles requièrent, d'une part, la connaissance de l'expression analytique des fonctions, et d'autre part le calcul des dérivées, la continuité, linéarité, etc. Toutes ces raisons mènent à considérer ces fonctions comme des *boîtes-noires*. Ce concept (boîte-noire) permet de faire abstraction de la fonction dont l'exploitation se fait uniquement par l'analyse de la réponse à une entrée donnée.

L'optimisation de boîtes-noires est appliquée dans des disciplines de plus en plus variées. Audet *et al.* (2010) formulent le problème d'optimisation des paramètres algorithmiques comme un problème de boîtes-noires visant à trouver les paramètres maximisant la performance de l'algorithme tel que le nombre total des évaluations et les temps de calcul. Le problème de prévention des risques d'un tsunami où il s'agit de trouver le positionnement des capteurs de détection maximisant le temps d'avertissement aux villes côtières de Spillane *et al.* (2009) est aussi modélisé comme un problème de boîtes-noires. Une autre application des boîtes-noires dans le domaine de l'hydroélectricité est décrite par Alarie *et al.* (2013) pour le problème de localisation de stations de mesure du couvert nival. Dans le problème d'optimisation de formes d'ailes de Marsden (2004), qui minimisent le bruit aérodynamique. En médecine, Yang *et al.* (2010) utilisent un modèle de boîte-noire pour l'amélioration des opérations chirurgicales des enfants atteints de malformations cardiaques.

Un problème d'optimisation de boîtes-noires sans contraintes est donnée par

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{1.1}$$

où la boite-noire f est la fonction objectif à minimiser qui prend ses valeurs de $\mathbb{R}^n$ dans $\mathbb{R} \cup \{+\infty\}$ en incluant le cas où les évaluations peuvent échouer. Par ailleurs, dans le cas de l'optimisation sous contraintes, ces dernières sont considérées comme des boîtes-noires.

Plusieurs algorithmes ont été conçus pour résoudre les problèmes de boîtes-noires dont l'algorithme MADS (*Mesh Adaptive Direct Search*) de Audet et Dennis (2006). C'est un algorithme de recherche directe utilisant uniquement la valeur de la fonction objectif évaluée à des points sur un treillis. Cet algorithme est discuté en détail à la section 2.2.3.

Le concept général de l'optimisation de boîtes-noires, illustré à la figure 1.1, consiste à évaluer la boîte-noire f à des points candidats générés par un algorithme d'optimisation de boîtes-noires puis, la valeur de f est renvoyée à l'algorithme afin de l'analyser et de générer d'autres candidats.

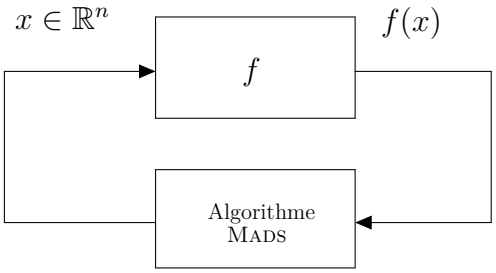


Figure 1.1 Schéma d'optimisation de boîte-noire.

1.1 Problématique de la recherche

Le problème traité dans ce projet s'inscrit dans le cadre de l'optimisation sans contraintes, défini par (1.1), où la boîte-noire à la particularité de retourner, dans certains cas, des valeurs différentes pour les mêmes valeurs d'une variable donnée en entrée. Ceci est dû à plusieurs paramètres intervenant dans la structure interne de la boîte-noire, comme la graine aléatoire ou le type de la machine. À la figure 1.2, nous considérons un exemple où la sortie $z_p(x)$ retournée par la boîte-noire z dépendant d'un seul paramètre qui est la graine aléatoire p. Si la graine est fixée a priori par l'utilisateur, nous parlerons de sortie déterministe. Par contre, si nous considérons p comme faisant partie de la boîte-noire, et donc inaccessible et incontrôlable, nous parlerons alors d'une sortie $f(x)$ stochastique.

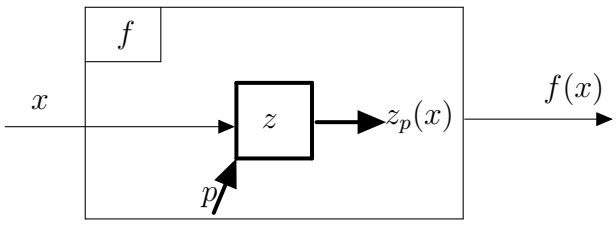


Figure 1.2 Les différentes sorties d'une boîte-noire.

Notre travail s'articule essentiellement autour de l'optimisation de boîtes-noires où la fonction est contaminée par un bruit à caractère stochastique qui peut être présent autant sur les entrées que sur la sortie $f(x)$. Dans une telle situation, l'optimisation ne dépend pas seulement de l'entrée x mais aussi de la variation de x autour d'une valeur nominale et un bruit affectant la sortie. Cette variation peut avoir un impact sur la solution de l'optimisation. En effet, il a été montré dans les travaux de Ben-Tal et Nemirovski (2000) que dans le cas contraint, une faible perturbation de la solution optimale affecte sa réalisabilité. Aussi, un autre problème peut survenir lorsqu'il existe une tolérance sur les variables de conception, tel que montré par Lee et Park (2001) pour le problème de construction d'une poutre où la valeur de l'objectif devient très sensible à de petites variations de la solution optimale dans l'intervalle de tolérance.

En théorie, la solution obtenue en utilisant les algorithmes d'optimisation classiques est celle qui donne la plus petite valeur de l'objectif. Or, en pratique cette solution est indésirable car sa variation entraine une détérioration drastique de la valeur de la fonction objectif. Pour mieux illustrer la problématique, la figure 1.3 montre que le minimum global x_1^* n'est pas la solution désirée en industrie étant donné que les ingénieurs ont souvent recours à l'ajustement d'une telle solution selon leur besoin. Ainsi, même si la solution ajustée $x_1^* - \epsilon$, pour une petite valeur de ϵ, est dans un voisinage proche du minimum global obtenu, le bruit présent dans la fonction fait en sorte que la valeur de l'objectif s'éloigne beaucoup de la valeur optimale, $f(x_1^* - \epsilon) \gg f(x_1^*)$. De ce fait, la solution que nous voulons obtenir, et que nous qualifierons de robuste est la solution ayant le plus large voisinage dans lequel l'évaluation de la fonction donne des valeurs très proches de la valeur optimale. Une telle solution est illustrée par la solution x_2^*.

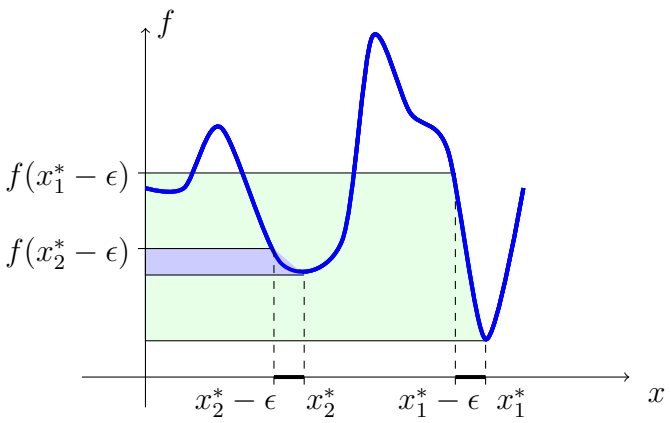


Figure 1.3 La solution x_1^* est un minimum globale de la fonction f et la solution x_2^* est un minimum robuste.

Le but de notre travail de recherche est de proposer un algorithme donnant une solution robuste au bruit qui possède une bonne valeur de la fonction objectif. Cette solution donne un bon compromis entre robustesse et optimalité. Cet algorithme utilise l'algorithme MADS que nous présenterons à la section 2.2.3.

1.2 Plan du mémoire

Ce mémoire est organisé comme suit. Le chapitre 2 est consacré à la revue de littérature où on y trouve d'abord un historique des méthodes d'optimisation sans dérivées dont deux grandes familles d'algorithmes pour l'optimisation de boîtes-noires. La première se base sur l'optimisation d'un modèle de la boîte-noire construit localement et la deuxième utilise des recherches directes et se base sur des directions, dont l'algorithme MADS fait partie. Dans ce chapitre nous présentons également un bilan des travaux actuels et passés sur l'optimisation sous incertitude et l'optimisation robuste en particulier.

L'objectif du chapitre 3 est d'exposer l'approche algorithmique proposée pour palier à la problématique énoncée à la section 1.1. Cette approche consiste à trouver un algorithme capable de réduire le bruit dans les boîtes-noires et retourner des solutions robustes. L'approche proposée est également accompagnée d'une analyse de convergence. Nous présentons au chapitre 4 différentes façons permettant d'évaluer la robustesse des solutions obtenues par l'application de l'algorithme proposé. Ces solutions sont comparées aux solutions retournées par MADS afin de valider notre approche.

Finalement, nous terminons ce mémoire par une conclusion qui aboutit à des points de départ de plusieurs pistes de recherche afin d'étendre les travaux présentés dans ce mémoire.

CHAPITRE 2

REVUE DE LITTÉRATURE

Ce chapitre traite des différentes méthodes d'optimisation sans dérivées, développées à ce jour, en mettant l'accent sur les méthodes de recherche directe directionnelle et plus particulièrement l'algorithme MADS sur lequel repose ce travail, ainsi qu'un historique sur les méthodes ayant mené à ce dernier. Par la suite, d'autres types de problèmes, qui se rapprochent au problème traité dans ce projet de recherche, sont introduits. Soient les problèmes d'optimisation sous incertitude et en l'occurrence les problèmes d'optimisation robuste.

2.1 Méthodes d'optimisation sans dérivées

Les méthodes d'optimisation sans dérivées ne requièrent pas d'information sur les dérivées et utilisent uniquement les valeurs de la fonction et des contraintes du problème à optimiser considérant qu'elles sont l'unique information connue pour le problème traité. En effet, l'estimation des dérivées est souvent très coûteuse ou difficile à cause de la particularité des fonctions. L'application de ces méthodes a augmenté exponentiellement au cours des dernières années avec l'accroissement de la complexité des fonctions définissant des cas réels en industrie. On peut noter à titre d'exemple les problèmes traités par Kokkolaras *et al.* (2001) et par Abramson (2004) qui consistent à optimiser la composition dans un système d'isolation thermique, et Marsden *et al.* (2004), qui s'articulent autour de l'optimisation de formes d'ailes et autour de la minimisation du bruit aérodynamique dans (Marsden (2004)).

Plusieurs méthodes d'optimisation sans dérivées ont été développées. Il en existe principalement trois. Il y a celles qui effectuent des opérations sur un simplexe, comme la méthode de Nelder et Mead (1965). Il y a aussi les méthodes se basant sur des directions pour guider l'optimisation vers un optimum local, que nous développerons à la section suivante. De plus, on trouve les méthodes qui utilisent un modèle pour approcher la fonction objectif dans une région de confiance, comme décrites par Powell (2003) ou encore par Conn *et al.* (2009).

L'optimisation par région de confiance est l'une des classes de méthodes d'optimisation sans dérivées les plus étudiées. Elle est conçue pour des problèmes non linéaires où la fonction objectif et les contraintes n'ont pas les caractéristiques de boîtes-noires. Par ailleurs cette classe de méthodes suppose l'existence de dérivées bien qu'elle ne les utilise pas. Dans le cas sans contraintes, elle se base sur la minimisation d'un modèle quadratique qui approche la fonction objectif à chaque itération dans une région de confiance $B(x_k, \Delta_k)$, où Δ_k est le

rayon de la boule centrée au point courant x_k. L'utilisation d'un modèle quadratique permet un bon traitement de l'information reliée à la courbure de la fonction objectif. Dans la même optique, Powell (2002) a introduit l'algorithme UOBYQA pour la résolution des problèmes sans dérivées de faible dimension et non contraints, en utilisant l'algorithme de région de confiance. Plus tard, il développe l'algorithme NEWUOA dans Powell (2006), basé sur la méthode de région de confiance et suggère d'utiliser un nombre réduit de points d'interpolation, soit $m \in [n+2, (n+1)(n+2)/2]$ où n est la taille du problème, pour la construction du modèle d'interpolation quadratique. Puis, Powell (2010) revient sur la preuve de convergence de la méthode. Aussi, il propose une nouvelle façon de faire la mise à jour du modèle dans Powell (2013). On trouve aussi d'autres algorithmes utilisant les région de confiance comme l'algorithme WEDGE de Marazzi et Nocedal (2002).

L'algorithme DIRECT pour *DIviding RECTangles*, introduit par Jones *et al.* (1993), est un algorithme itératif conçu pour les problèmes avec contraintes de bornes. Il repose sur la recherche des solutions dans une partition du sous-espace de recherche appelé hyper-rectangle. Chaque itération débute par la recherche d'hyper-rectangles optimaux susceptibles de contenir un optimum global. Cette étape est poursuivie par la division de ces hyper-rectangles en plus petits sous-espaces afin d'évaluer la fonction au centre de chaque hyper-rectangle. Plus précisément, un hyper-rectangle est dit potentiellement optimal s'il satisfait la définition suivante :

Définition 1. *Supposons que nous avons à partitioner un hypercube en l hyper-rectangles. Soit c_i et d_i le centre et la taille du i-ème hyper-rectangle. Un hyper-rectangle j est potentiellement optimal s'il existe une constante T positive telle que :*

$$\begin{aligned} &f(c_j) - Td_j \leq f(c_i) - Td_i, \qquad \forall i = 1, ..., l, \\ &f(c_j) - Td_j \leq f_{min} - \eta \mid f_{min} \mid, \end{aligned} \tag{2.1}$$

où η une constante positive.

2.2 Méthodes de recherche directe directionnelle

Dans cette section sont illustrées les méthodes de recherche directe. Elles sont appelées ainsi car elles ne supposent aucune condition sur la fonction objectif et les contraintes comme la convexité, la continuité, ou la différentiabilité, par exemple. Ces méthodes considèrent que les dérivées sont inaccessibles ou inestimables et elle utilisent uniquement les valeurs de la fonction pour conduire l'optimisation. Elles ont été conçues pour les problèmes de type boîte-noire et elles reposent sur la génération de directions qui permettent d'explorer l'espace de recherche. Le premier algorithme de recherche directe à avoir vu le jour est l'algorithme de

recherche par coordonnées (Cs : Coordinate Search) de Fermi et Metropolis (1952) dont les directions de recherche sont les vecteurs de la base canonique de $\mathbb{R}^n$. Une évolution de (Cs) mène à l'algorithme de recherche par motifs (Gps : Generalized Pattern Search) présenté par Torczon (1997), qui utilise des directions de recherche plus élaborées. Finalement, ce dernier a été généralisé par Audet et Dennis (2006) pour donner naissance à l'algorithme de recherche sur treillis adaptifs (Mads : Mesh Adaptive Direct Search), qui propose notamment une méthode encore plus perfectionnée pour la génération des directions de recherche ainsi qu'une rigoureuse analyse de convergence. Une mine d'informations sur les méthodes de recherche directe de l'optimisation de boîte-noire et leurs applications sont résumées dans l'article Audet (2014). Avant d'expliquer ces algorithmes en détails, il convient d'abord d'introduire la notion d'ensemble générateur positif, qui est utilisée pour la définition des directions.

Base positive

La notion de base positive a été introduite par Davis (1954). Une brève définition telle qu'elle a été définie dans l'article de Audet (2011) est illustrée ici.

Un ensemble fini de vecteurs $\mathcal{D} = \{d_1, d_2, ..., d_p\} \subset \mathbb{R}^n$ est un ensemble générateur positif pour $\mathbb{R}^n$, si chaque vecteur $v \in \mathbb{R}^n$ peut s'exprimer par une combinaison linéaire positive des vecteurs de $\mathcal{D}$. L'ensemble $\mathcal{D}$ forme une base positive pour $\mathbb{R}^n$ s'il est un ensemble générateur minimal, i.e., aucun sous-ensemble de $\mathcal{D}$ ne peut générer positivement $\mathbb{R}^n$. La figure 2.1 montre des exemples de bases positives.

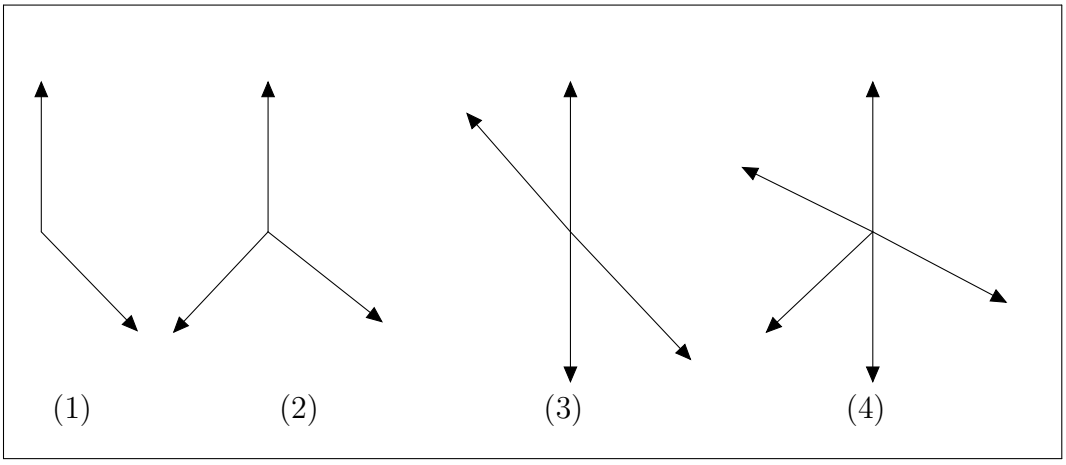


Figure 2.1 Exemples de bases et d'ensembles générateurs positifs dans $\mathbb{R}^2$.

(1) n'est pas une base positive. (2), (3) et (4) sont des ensembles générateurs positifs. (2) et (3) sont des bases positives. Extrait de l'article de Audet (2011).

2.2.1 Algorithme de recherche par coordonnées

La recherche par coordonnées est un algorithme utilisé pour la résolution des problèmes sans contraintes qui s'écrivent sous la forme :

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.2}$$

Chaque itération k débute par l'évaluation de la fonction f à des points d'essai générés autour de l'itéré courant x_k à un pas de longueur Δ_k^m suivant les $2n$ directions $\{\pm e_i\}_{i=1}^n$ dont n sont les directions de la base usuelle $\mathbb{R}^n$ et les n directions opposées. Cette exploration des points d'essai dans l'espace discrétisé, appelé *treillis*, a pour but de rechercher un point ayant une plus petite valeur de l'objectif que celle de l'itéré courant. Cette recherche est appelée *sonde locale* et l'itéré courant est désigné par *centre de sonde.* L'ensemble constituant les points de sonde est défini par :

$$P_k \quad = \quad \{x_k \pm \Delta_k^m e_i, i = 1, ...n\}, \tag{2.3}$$

où Δ_k^m est la taille du treillis. Cette notation est la même que celle adoptée dans Mads, présenté ultérieurement. Si une meilleure solution est trouvée dans P_k, l'itération est un succès et le prochain itéré devient ce point trouvé. Dans le cas où l'itération est un échec, c'est-à-dire : $\forall x \in P_k$, $f(x) \geq f(x_k)$, la taille du treillis est réduite de moitié afin d'évaluer f à des points plus proches de x_k. La recherche d'un meilleur point se fait jusqu'à ce que la taille du treillis Δ_k^m s'approche de la taille minimale, ou que le nombre maximal des évaluations de f est atteint. Cette méthode est très facile à implémenter néanmoins elle présente plusieurs lacunes. En effet, puisque la recherche se fait suivant un nombre restreint de directions, qui sont les mêmes à chaque itération, des régions de l'espace potentiellement intéressantes risquent de ne jamais être visitées. D'autre part, la recherche se fait uniquement de manière locale et aussi, même si un meilleur point a été trouvé parmi l'ensemble P_k, la fonction est évaluée à tous les autres points de P_k au lieu d'arrêter les évaluations au premier succès. Plus loin nous allons voir que cette technique peut évoluer en arrêtant les évaluations dès le premier succès, ce qui s'appelle *la recherche opportuniste.*

Les algorithmes suivants sont des évolutions de (Cs) et ils ont été développés pour combler ces lacunes.

2.2.2 Algorithme de recherche par motifs

L'algorithme de recherche par motifs (Gps) a été conçu originellement pour la résolution des problèmes sans contraintes puis étendu à d'autres types de problèmes. Ces problèmes

sont : les problèmes avec contraintes de bornes par Lewis et Torczon (1999) ; contraintes linéaires de Lewis et Torczon (2000) ; contraintes générales introduit par Audet et Dennis, Jr. (2004) ; les problèmes à variables catégoriques dans l'article de Audet et Dennis (2001).

Cet algorithme permet d'effectuer une sonde locale suivant un ensemble fini de directions diversifiées plutôt que les $2n$ directions de la base usuelle. À chaque itération k, l'ensemble des directions $D_k = \{d_k^1, d_k^2, ..., d_k^{p_k}\}$ et D, l'ensemble des directions possibles, doivent satisfaire les conditions suivantes :

— D_k représente un ensemble générateur positif et $D_k \subseteq D$;

— Chaque direction $d_j \in D$, $j = 1, ..., n_D$, doit être issue d'un produit d'une matrice non singulière $G \in R^{n \times n}$ par un vecteur d'entiers $z_j \in \mathbb{Z}^n$.

Les points de la sonde suivant ces directions sont définis par :

$$P_k \quad = \quad \{x_k + \Delta_k^m d : d \in D_k\}. \tag{2.4}$$

À l'itération k, l'ensemble V_k contient l'historique des points où la fonction f a été évaluée depuis le début de l'itération. Tous les itérés générés doivent appartenir au treillis, qui est une discrétisation de l'espace des variables, défini par :

$$M_k \quad = \quad \{x + \Delta_k^m Dz : x \in V_k, z \in \mathbb{N}^p\}, P_k \subset M_k. \tag{2.5}$$

De plus, avant l'étape de sonde, l'algorithme propose une étape optionnelle appelée la recherche globale (SEARCH). À cette étape, la fonction objectif est évaluée à un ensemble de points générés sur le treillis dans le but de trouver un meilleur point que l'itéré courant x_k, i.e., la meilleure solution trouvée depuis le début de l'exécution de l'algorithme. La recherche globale est flexible car elle donne la possibilité à l'utilisateur de choisir des solutions qu'il juge intéressantes pour le type de problème traité. Il peut aussi asseoir ses propres stratégies de recherche ou des stratégies plus génériques comme la recherche aléatoire. Un exemple de combinaison de la stratégie de recherche à voisinage variable avec la recherche globale peut être trouvé dans Audet *et al.* (2008a).

Contrairement à la recherche par coordonnées, GPS offre, en plus de la recherche globale et des directions plus diversifiées, la possibilité d'augmenter la taille du treillis quand une itération est qualifiée de succès, telle que, pour $\tau \in \mathbb{Q}$, w^+ et w^- des entiers, on a

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m, \quad \text{avec} \tag{2.6}$$

$$w_k = \begin{cases} \{0, 1, ..., w^+\} & \text{si l'itération est un succès,} \\ \{w^-, w^- + 1, ..., -1\} & \text{sinon.} \end{cases}$$

La possibilité d'augmenter Δ_k^m permet d'améliorer considérablement la vitesse de convergence. De plus, la recherche se fait selon les directions de succès de l'itération précédente. En effet, les directions sont classées de sorte que celles qui ont mené à un succès sont placées en tête de la liste des prochaines directions.
On peut remarquer que dans le cas particulier de la recherche par coordonnées (algorithme CS), l'ensemble des directions D est $[I_n - I_n]$, $\tau = 2, w^- = -1$, et $w^+ = 0$. Ainsi, si l'itération est un succès, la taille du treillis est inchangée ($\Delta_{k+1}^m = \tau^0 \Delta_k^m$), sinon elle est réduite par deux ($\Delta_{k+1}^m = \tau^{-1} \Delta_k^m$).

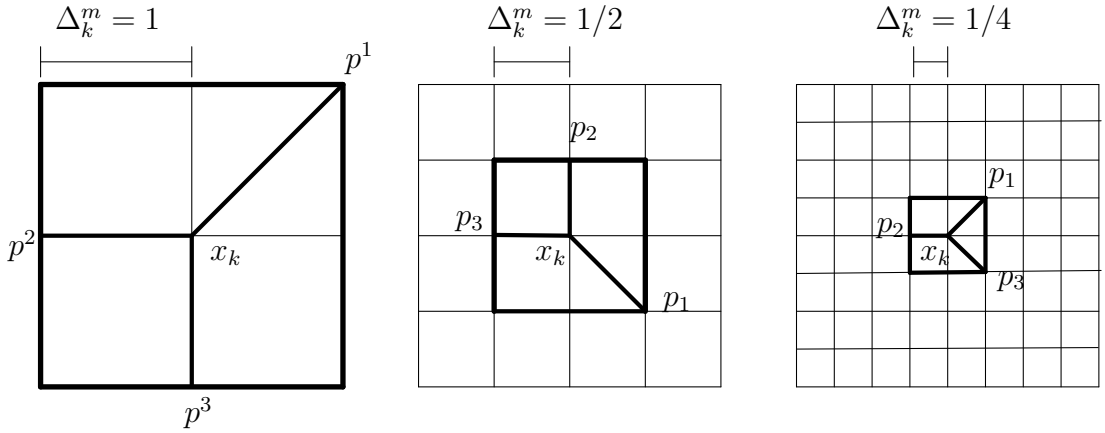


Figure 2.2 Exemple de différents cadres de Gps : $P_k = \{x_k + \Delta_k^m d : d \in D_k\} = \{p_1, p_2, p_3\}$. Extrait de l'article Audet et Dennis (2006).

L'utilisation des bases positives présente un grand intérêt pour la construction des directions de recherche. En effet, le théorème énoncé par Lewis et Torczon (1996) et repris par Abramson *et al.* (2004) montre que si $\mathcal{D} = \{d_1, d_2, ..., d_p\}, d_i \neq 0$, engendre $\mathbb{R}^n$ linéairement, alors pour chaque vecteur v non nul dans $\mathbb{R}^n$, il existe un indice i pour lequel $v^T d_i > 0$, c'est-à-dire qu'il existe au moins une direction dans chaque demi-espace. De plus, si la dérivée existe en un point x et $\nabla f(x) \neq 0$, en prenant $v = -\nabla f(x)$, il existe un indice i tel que $\nabla f(x)^T d_i < 0$ et on peut affirmer qu'il existe au moins une direction de descente dans $\mathcal{D}$.

2.2.3 Algorithme de recherche par treillis adaptifs

L'algorithme Mads : *Mesh Adaptive Direct Search* appartient à la classe des algorithmes de recherche directe avec et sans contraintes et vient généraliser l'algorithme Gps. Il est

décrit ici tel qu'il a été évoqué dans Audet et Dennis (2006). À l'étape (SEARCH) de chaque itération la fonction f est évaluée pour un ensemble fini de points générés sur le treillis. Ces valeurs sont comparées à la valeur de la fonction au point courant x_k. Parmi les points du treillis, le premier donnant une valeur de l'objectif inférieure à celle de la solution courante est retenu. L'itération prend donc fin suivant la stratégie opportuniste et est considérée comme un succès. Ainsi, l'itération suivante débute par une mise à jour de la nouvelle solution courante, x_{k+1} telle que $f(x_{k+1}) < f(x_k)$, et un accroissement de la taille du treillis $\Delta_k^m \leq \Delta_{k+1}^m$ afin d'accélérer la convergence. Par contre, un échec de l'étape SEARCH conduit à la deuxième étape appelée sonde locale (POLL) juste avant la fin de l'itération. Cette étape consiste en l'exploration locale de l'espace des variables autour de la solution courante dans des directions choisies avec plus de flexibilité. L'ensemble des points de sonde est dans un cadre de taille Δ_k^p avec $\Delta_k^m \leq \Delta_k^p$. Le cadre est défini par

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k. \tag{2.7}$$

Cependant, ce qui distingue MADS de GPS est que MADS considère un paramètre supplémentaire, Δ_k^p, qui représente la taille du cadre tandis que dans GPS, Δ_k^m représente à la fois la taille du treillis et du cadre. Soit $\Delta_k^m = \Delta_k^p$.

Dans le cas où l'itération se termine par un échec, la taille du treillis est réduite afin de réévaluer la fonction dans un espace plus proche de la solution courante. La taille du treillis est actualisée à l'itération suivante telle que $\Delta_{k+1}^m = \tau^{w_k}\Delta_k^m$, où w_k est défini par (2.6). Une illustration des deux paramètres de taille de treillis et du cadre est présentée à la figure 2.3.

Un autre point important à soulever est que ces paramètres doivent respecter les conditions suivantes :

- $\Delta_k^m \leq \Delta_k^p, \forall k$;
- $\lim_{k\in K} \Delta_k^m = 0 \Leftrightarrow \lim_{k\in K} \Delta_k^p = 0, \forall K$ un sous-ensemble fini d'indices d'itérations.

En d'autres termes, Δ_k^m est toujours inférieur à la taille du cadre ce qui offre plus de possibilités pour la construction des points de sonde. Avec GPS, plusieurs régions ne sont pas accessibles car le nombre de directions est fini. La figure 2.2 illustre des exemples de points de sonde possibles avec GPS. La considération des deux paramètres Δ_k^m et Δ_k^p dans MADS, permet de construire un ensemble dense de directions donnant la possibilité d'évaluer des points dans n'importe quelle direction à l'intérieur du cadre de sonde. À la figure 2.3 le cadre de sonde est représenté en gras.

Par ailleurs, l'ensemble des directions utilisées D_k n'est pas inclus dans l'ensemble des directions possibles D et pour chaque $d \in D_k$:

— d est une combinaison entière positive des directions de D, i.e., $\exists u \in \mathbb{N}^{n_{D_k}}$ tel que $d = Du$. Ainsi la condition (2.7) est respectée, et $\forall x \in P_k, x \in M_k$.

— La distance entre le centre de sonde x_k et les points de P_k est bornée. En effet, les points de sonde $(x_k + \Delta_k^m d)$ doivent être à l'intérieur du cadre, i.e.,

$$\begin{aligned} \|x_k + \Delta_k^m d - x_k\| &\leq c\Delta_k^p \text{ où } c \in \mathbb{R}, \\ \Leftrightarrow \Delta_k^m \|d\| &\leq \Delta_k^p \max\{\|d'\| : d' \in D\}, \end{aligned}$$

en choisissant $c = \max\{\|d'\| : d' \in D\}$.

— La limite des ensembles normalisés $D_k = \{\frac{d}{\|d\|} : d \in D_k\}$ sont des ensembles générateurs positifs. Ceci évite le cas où les directions choisies convergent vers une base qui n'est pas une base positive.

Ainsi, l'avantage de MADS est de permettre de générer un ensemble dense de directions de sonde D_k qui permet la convergence de la taille du treillis vers zéro plus rapidement que la taille du cadre Δ_k^p.

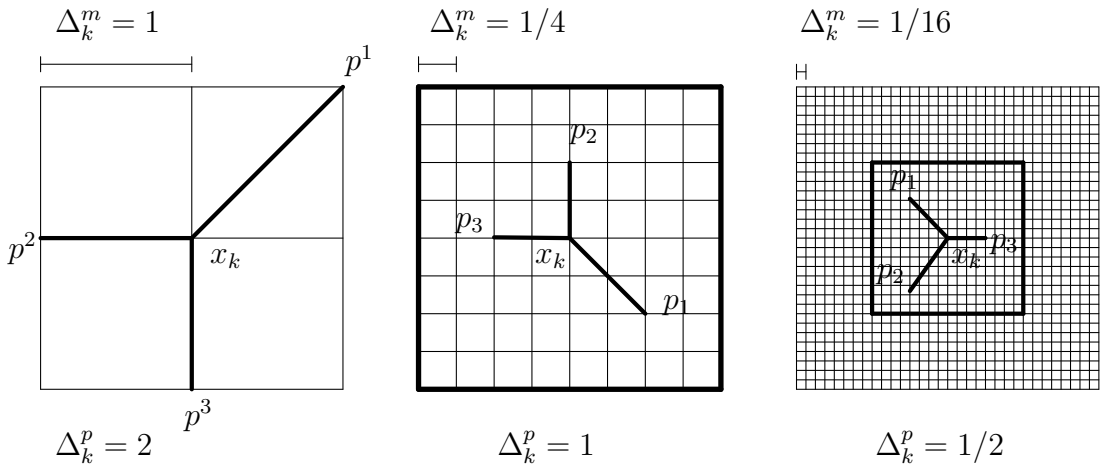


Figure 2.3 Exemple de différents cadres de MADS $P_k = \{x_k + \Delta_k^m d : d \in D_k\} = \{p_1, p_2, p_3\}$. Extrait de l'article de Audet et Dennis (2006).

Résultats de convergence

Les résultats de l'analyse de convergence introduits ici découlent directement de l'article Audet et Dennis (2006). Ils sont basés sur la différentiabilité locale de f et les propriétés de l'ensemble des contraintes Ω. Les hypothèses émises sont les suivantes : H1 : la valeur de f à un point de départ réalisable est finie ;
H2 : tous les itérés générés par MADS appartiennent à un ensemble compact.

Proposition 2. *La taille du treillis et du cadre satisfont la condition :*

$$\liminf_{k\to\infty} \Delta_k^p = \liminf_{k\to\infty} \Delta_k^m = 0.$$

Définition 3. *La sous-suite des centres de sonde $\{x_k\}_{k\in K}$, dont les itérations ont été qualifiées d'échec, est dite une sous-suite raffinante si $\{\Delta_k^p\}_{k\in K}$ converge vers zero, où K est un sous-ensemble d'indices. Soit $\hat{x}$ le point vers lequel la suite raffinante converge. Si $\lim_{k\in L} \frac{d_k}{\|d_k\|}$ existe pour $L \subseteq K$, $d_k \in D_k$, et si $x_k + \Delta_k^m d_k \in \Omega$, pour un nombre infiniment grand $k \in L$, alors cette limite est dite une direction raffinante pour $\hat{x}$.*

Sous les hypothèses H1 et H2, il existe au moins une suite raffinante convergente.

Théorème 4. *Sous les deux hypothèses H1 et H2, il existe au moins un point $\hat{x}$ vers lequel la suite x_k converge. Si f est semi-continue inférieurement près de $\hat{x}$, alors $\lim_{k\in K} f(x_k)$ existe et $\lim_{k\in K} f(x_k) \geq f(\hat{x})$.*

Nous définissions la dérivée directionnelle au sens de Clarke (1983) comme suit

Définition 5. *La dérivée généralisée de Clarke de f en $\hat{x}$ dans la direction $v \in \mathbb{R}^n$ est :*

$$f^\circ(\hat{x}; v) = \limsup_{x\to\hat{x}, t\downarrow 0} \frac{f(x+tv) - f(x)}{t}.$$

Muni de cette définition nous pouvons présenter le théorème principale de l'analyse de convergence de l'algorithme MADS.

Théorème 6. *Si $\hat{x}$ est la limite d'une suite raffinante, et si d est une direction dans D pour laquelle f a été évaluée un nombre infini de fois pour tous les itérés dans l'étape de sonde, et si f est Lipschitz près de $\hat{x}$ alors la dérivée généralisée de Clarke de f au point $\hat{x}$ dans la direction d est non négative : $f^\circ(\hat{x}; d) \geq 0$.*

2.3 Le logiciel NOMAD

NOMAD (*Nonlinear Optimization with the* MADS *Algorithm*) est un logiciel codé en c++ qui implémente l'algorithme MADS. Il est conçu spécifiquement pour l'optimisation sous contraintes de fonctions boîtes-noires de la forme

$$\min_{x\in\Omega} \quad f(x) \tag{2.8}$$

où $\Omega = \{x \in X \subset \mathbb{R}^n : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$ est l'ensemble réalisable, $f, c_j : X \to \mathbb{R} \cup \{\infty\}$ pour tout $j \in J = \{1, 2, ..., m\}$, et X regroupe les contraintes non relaxables et non quantifiables ainsi que les contraintes de bornes. La fonction f ne peut être évaluée en un point hors de X. Les contraintes $c_j(x) \leq 0, j \in J$, définissant l'ensemble Ω sont des contraintes relaxables et quantifiables, c'est-à-dire que non seulement l'évaluation de la fonction f peut avoir lieu pour un point x violant une de ces contraintes qui possèdent les caractéristiques d'une boîte-noire, mais encore elles mesurent de combien les contraintes sont violées.

Le logiciel NOMAD est largement utilisé aussi bien dans le milieu académique que par des industriels, comme Boeing et Exxon Mobil. Il a été développé par Abramson *et al.* (2014) et décrit par Le Digabel (2011). Les problèmes sans contraintes où $\Omega = \mathbb{R}^n$ sont considérés pour ce projet de recherche.

Certains paramètres de NOMAD

Stratégie de recherche

Plusieurs stratégies pour générer les points d'essai sont disponibles dans MADS : la recherche à voisinage variable (VNS : *Variable Neighborhood* SEARCH) de Mladenović et Hansen (1997), la recherche par Latin Hypercube (LH) par McKay *et al.* (1979) et la recherche spéculative introduite par Audet et Dennis (2006). Cette dernière est activée à la prochaine itération si l'itération courante est qualifiée de succès. La recherche spéculative consiste à évaluer la fonction à un point généré plus loin suivant la direction du dernier succès, ce qui permet d'augmenter la vitesse de convergence de l'algorithme. Cette stratégie de recherche est activée par défaut dans NOMAD. Le lecteur voulant en savoir davantage sur les deux autres stratégies est invité à consulter les articles cités en référence. Par ailleurs, les points générés par MADS au cours d'une itération ne sont pas tous évalués car l'itération s'arrête au premier succès. Cette stratégie est appelée la stratégie opportuniste et est activée par défaut dans NOMAD.

L'évaluation par blocs

L'algorithme MADS génère, au cours d'une itération, une liste de points auxquels f doit être évaluée. Dans la version originale de MADS, la fonction f était évaluée à un point à la fois puis l'évaluation est retournée une à la fois. Désormais, depuis la version NOMAD.3.6.2, il est possible d'envoyer un bloc de points à la boîte-noire. La taille maximale d'un bloc est égale à un par défaut dans NOMAD. La fonction permettant l'évaluation par blocs peut être adaptée par l'utilisateur afin d'évaluer les points d'un bloc parallèlement. Un tel exemple

peut être trouvé dans les exemples fournis avec NOMAD. Par ailleurs, il existe des extensions permettant l'évaluation en parallèle de plusieurs points d'essai, comme l'algorithme PSD-MADS présenté par Audet *et al.* (2008b) qui permet la résolution de problèmes avec un plus grand nombre de variables en procédant à une décomposition spatiale. Une liste exhaustive des extensions de MADS permettant de faire des évaluations en parallèle est donnée par Le Digabel *et al.* (2010). Leur utilisation réduit le temps d'exécution de façon significative.

Toutes ces méthodes d'optimisation sans dérivées ont été appliquées à des cas réels en industrie. Mais souvent, le système à optimiser est affecté par plusieurs paramètres incertains qui influencent significativement la solution. Afin de traiter cette problématique, plusieurs approches de traitement de l'incertitude ont été abordées dans la littérature. Certains travaux se basent sur l'optimisation robuste, et d'autres reposent sur l'amélioration ou la modification de méthodes afin de prendre en compte l'incertitude. La section suivante donne une vue globale de l'optimisation sous incertitude en définissant les différents problèmes incertains.

2.4 Traitement de l'incertitude

Il existe plusieurs types d'incertitudes qui peuvent être classées de différentes façons selon leurs sources. D'après Beyer et Sendhoff (2007), Ben-Tal et Nemirovski (2002) et Goerigk (2014) l'incertitude peut être reliée à :

- Des changements environnementaux : la solution d'un problème d'optimisation sur un horizon de planification à long terme peut être influencée par des facteurs extérieurs comme l'humidité, la température ou un changement des propriétés du matériel. Dans ce cas, la fonction à optimiser depend des variables de conception et des paramètres environnementaux ;
- Un paramètre de tolérance de production : dans ce cas l'incertitude est présente sur les variables de conception qui fluctuent autour d'une valeur nominale (poids, longueur, pureté du produit, etc). Comme c'est le cas en métallurgie dans le procédé de fabrication de plaques stratifiées (voir Walker et Hamilton (2005)) et aussi pour le problème de conception d'une structure d'une poutre (voir Lee et Park (2001)) ;
- Une imprecision dans l'évaluation : celle-la est causée par des erreurs de mesures ou d'implementation qui surviennent dans un cas réel quand une approximation de la fonction est utilisée au lieu de la vraie fonction. Dans ce cas les données utilisées sont construites par échantillonnage. Aussi, quand la fonction est issue d'un code informatique contenant des simulations. Un exemple de ce type d'incertitude peut être trouvé dans les travaux de Ciccazzo *et al.* (2013) ;

- La préférence d'une décision à une autre : dans les problèmes où il s'agit de trouver la décision optimale d'investissement de portefeuille, la prise de décision est reliée aux données du marché et fondamentalement aux préférences des investisseurs qui présentent une source d'incertitude qui est traitée par Armbruster et Delage (2014).

2.4.1 L'optimisation robuste

Dans le cas où les fonctions sont bruitées, l'optimisation robuste permet de fournir une solution adéquate minimisant l'effet de la variation des entrées sur les sorties. Les premiers travaux sur l'optimisation robuste sont marqués par Soyster (1973).

Nous introduisons le concept général de l'optimisation robuste tel qu'énoncé par Goerigk (2014). Il consiste à inclure la variable incertaine dans le modèle. Ceci revient à reformuler le problème en un problème incertain $P(\xi)$ défini par,

$$\begin{array}{lll} (P(\xi)) & \min\limits_{x} & f(x,\xi) \\ & s.c. & F(x,\xi) \leq 0, \\ & & x \in X, \\ & & \xi \in \mathcal{U}, \end{array} \tag{2.9}$$

où $F(.,\xi) : \mathbb{R}^n \to \mathbb{R}^m$, $f(.,\xi) : \mathbb{R}^n \to \mathbb{R}$ et $X \subset \mathbb{R}^n$, où $\xi \in \mathcal{U} \subset \mathbb{R}^M$ est le paramètre incertain ou le bruit. La dimension de ξ pour un problème linéaire incertain $\min\limits_{x \in R^n}\{c^t x : Ax \leq b, (A,b,c) \in \mathcal{U}\}$ est la somme des dimensions de la matrice des contraintes, du vecteur du second membre et le vecteur coût.

La résolution de $(P(\xi))$ passe par la résolution du problème équivalent robuste dont la formulation est en général de type min max, minimisant le pire cas. Cette notion n'est pas récente. En effet, des travaux remontant à 1945 utilisent ce concept en finance dont il est souvent question de minimiser le pire risque comme dans l'article de Wald (1945). Ainsi, une solution optimale x est dite robuste si elle reste réalisable pour toutes les valeurs de ξ dans l'ensemble $\mathcal{U}$, autrement dit $F(x,\xi) \leq 0, \forall \xi \in \mathcal{U}$. Le problème équivalent robuste est donné par

$$\begin{array}{lll} (R) & \min\limits_{x} & \sup\limits_{\xi \in \mathcal{U}} f(x,\xi) \\ & s.c. & x \in R(\mathcal{U}), \\ & & x \in X, \end{array} \tag{2.10}$$

où $R(\mathcal{U}) = \bigcap\limits_{\xi \in \mathcal{U}}\{x \in X : F(x,\xi) \leq 0\}$ est l'ensemble des points réalisables pour n'importe

quelle valeur de ξ. Goerigk (2014), développeur de la librairie ROPI pour l'optimisation robuste, a également énuméré les différents logiciels d'optimisation robuste.

Ben-Tal et Nemirovski (2002) traitent différentes instances d'un problème convexe incertain créées en caractérisant l'ensemble d'incertitude. Ainsi, ils résolvent des problèmes linéaires, quadratiques puis semi-définis positifs incertains.

Une autre approche de formulation des problèmes sous incertitude en théorie de la décision consiste à intégrer la préférence du décideur interprétée par la fonction d'utilité. Armbruster et Delage (2014) proposent dans ce contexte plusieurs reformulations d'un problème incertain. Sous certaines conditions les auteurs se ramènent à des problèmes qui sont résolus en temps polynomial. La fonction d'utilité est définie par $\nu : \mathbb{R} \to \mathbb{R}$. Le scénario ξ_1 domine ou est préféré à ξ_2 et on note $\xi_1 \succeq \xi_2$, si et seulement si

$$\mathbb{E}[\nu(\xi_1)] \geq \mathbb{E}[\nu(\xi_2)], \tag{2.11}$$

où la fonction inconnue ν appartient à l'ensemble $\mathcal{E}$. Soit un problème stochastique déterminant la decision $x \in X$ qui maximise l'utilité sous le scénario aléatoire ξ :

$$\max_{x \in X} \quad \mathbb{E}[\nu(h(x, \xi))], \tag{2.12}$$

où h est la valeur monétaire de la decision x pour le scénario ξ. Trois formulations sont proposées :
La première est basée sur la maximisation du pire cas de la fonction d'utilité, soit :

$$\max_{x \in X} \inf_{\nu \in \mathcal{E}} \quad \mathbb{E}[\nu(h(x, \xi))]. \tag{2.13}$$

La deuxième consiste à reformuler le problème en un problème équivalent certain robuste qui s'écrit comme :

$$\max_{x \in X} \inf_{\nu \in \mathcal{E}} \quad \mathbb{C}_\nu[h(x, \xi)], \tag{2.14}$$

où $\mathbb{C}_\nu[\chi] := \sup\{s : \nu(s) \leq \mathbb{E}[\nu(\chi)]\}$, s et χ deux variables aléatoires, en considérant $h(x, \xi)$ une loterie, ce problème donne le plus grand montant d'argent pour lequel le décideur est prêt à échanger contre $h(x, \xi)$. Finalement, un problème d'optimisation stochastique avec une contrainte de domination tel qu'expliqué en (2.11)

$$\begin{aligned} & \max f(x) \\ s.c. \quad & \mathbb{E}[\nu(h(x, \xi))] \geq \mathbb{E}[\nu(h(\chi))] \ \forall \nu \in \mathcal{E}. \end{aligned} \tag{2.15}$$

Dans leurs travaux, la robustesse se définie par la meilleure solution par rapport à la pire utilité ν dans $\mathcal{E}$.

2.4.2 Optimisation des fonctions bruitées

Plusieurs travaux sont menés afin de développer des algorithmes d'optimisation où un bruit déterministe ou stochastique est présent dans la fonction. Dès lors, il convient de définir la nature du bruit afin d'identifier les méthodes adéquates pour son traitement. On peut trouver dans les travaux de Moré et Wild (2009) qu'un problème déterministe est caractérisé par la présence d'une composante déterministe désignant le bruit, soit

$$f(x) = (1 + \epsilon_f \phi(x)) \sum_{i=1}^{r} z_i(x)^2), r \in \mathbb{N} \tag{2.16}$$

où ϵ_f est un scalaire représentant le niveau de bruit et $\phi : \mathbb{R}^n \to [-1; 1]$ est la fonction bruit définie par le polynôme de Chebyshev T_3 telle que :

$$\phi(x) = T_3(\phi_o(x)), \qquad T_3(\alpha) = \alpha(4\alpha^2 - 3),$$

et $\phi_0(x) = 0,9 \sin(100\|x\|_1) \cos(100\|x\|) + 0,1 \cos(\|x\|_2)$. Ici le polynôme de Chebyshev est utilisé pour éliminer la périodicité dans la fonction ϕ. Quant aux problèmes stochastiques, la fonction ϕ est remplacée par un générateur de nombres aléatoires ou comme défini par Moré et Wild (2011) :

$$f(x) = z(x) + \epsilon \tag{2.17}$$

où $z : \mathbb{R}^n \to \mathbb{R}$ est une fonction lisse et le bruit ϵ est une variable aléatoire dont la distribution est indépendante de x.

Un des plus récents travaux traitant le bruit est décrit par Larson et Billups (2013). Ils développent un algorithme de région de confiance qui fournit un point stationnaire de premier ordre pour le modèle, en ayant seulement accès aux valeurs de la fonction objectif contaminée par le bruit. Les problèmes traités sont de la forme (2.17), où le bruit ϵ est indépendant et identiquement distribué selon une loi normale de moyenne 0 et de variance $\sigma^2 < \infty$ et z une fonction suffisamment lisse. Dans leur algorithme, le modèle quadratique permettant d'approcher la fonction objectif est construit par régression en considérant toutes les évaluations précédentes.

Dans l'article de Liu (2013), il est montré que l'algorithme DIRECT de Jones *et al.* (1993), cité à la section 2.2, est très sensible à la mise à l'échelle additive de la fonction objectif et

insensible à l'échelle multiplicative. Cela signifie que, dans un contexte de minimisation des problèmes avec contraintes de bornes, la solution du problème $af(x)+b$ où $b \in \mathbb{R}$ est sensible quand b varie. La modification dans le choix de l'hyper-rectangle optimal permet d'obtenir l'algorithme DIRECT-a qui est plus robuste dans le sens où la valeur de l'objectif n'est pas affectée par les variations de b. En d'autres termes, la valeur de la solution est identique à celle de la solution du problème avec $(a = 1, b = 0)$. La transformation de l'algorithme réside dans le choix de l'hyper-rectangle optimal et l'équation (2.1) devient alors :

Définition 7.

$$\begin{aligned} &f(c_j) - Td_j \leq f(c_i) - Td_i, && \forall i = 1, ..., l, \\ &f(c_j) - Td_j \leq f_{min} - \mid f_{min} - \eta f_{moy} \mid, \end{aligned}$$

où f_{moy} est la moyenne des évaluations précédentes en retirant les valeurs extrêmes (maximale et minimale). Une année après, Liu *et al.* (2014) ont combiné cette version robuste de l'algorithme avec un algorithme multi-grille de Briggs *et al.* (2000) pour donner naissance à l'algorithme MrDIRECT *Multilevel robust DIRECT*.

Elster et Neumaier (1995) ont réutilisé la méthode de région de confiance pour traiter les problèmes bruités de faible dimension avec contraintes de bornes. Au cours des itérations, l'espace de recherche est réduit au cours de l'optimisation et le modèle quadratique, construit sur un espace à petite échelle, devient de plus en plus robuste car l'influence du bruit diminue. Dans le cadre de leur travail, la robustesse est mesurée par le nombre d'évaluations nécessaires pour aboutir à une solution donnant une valeur proche de l'optimalité. Cet algorithme a été comparé à l'algorithme Nelder-Mead (NM) qui a été modifié afin de prendre en compte les contraintes de bornes. Leur algorithme est significativement plus rapide que (NM).

Également, Deng et Ferris (2006) développent une variante de l'algorithme UOBYQA dans Powell (2002) pour le traitement des problèmes stochastiques de la forme

$$\min_{x \in \mathbb{R}^n} \quad f(x) = \mathbb{E}[z(x, \epsilon_x)], \tag{2.18}$$

où f est une fonction inconnue à estimer et z est la fonction affectée par le facteur bruit ϵ_x au point x. En effet, le bruit présent dans la fonction affecte le modèle quadratique construit dans la région de confiance dans le sens où sa minimisation retourne des solutions indésirables. Afin de palier à ce problème ils proposent de réduire la variance du modèle quadratique. Pour ce faire, les paramètres de ce modèle sont calculés en considérant la moyenne de plusieurs évaluations de la fonction en un point. Le nombre d'évaluations requis pour obtenir une bonne estimation de la valeur de la fonction objectif est déterminé par une technique de simulation de Bayes énoncé par Chick *et al.* (2001). Un an plus tard, Deng et Ferris (2007) se sont

intéressés au traitement du bruit dans l'algorithme DIRECT. Ils ont utilisé la même technique de détermination du nombre d'évaluations nécessaire pour une bonne approximation de la valeur de la fonction objectif dans le but de trouver un hyper-rectangle potentiellement intéressant.

Une autre technique d'optimisation d'un système stochastique est décrite par Sriver *et al.* (2009). Elle repose sur l'utilisation d'une procédure statistique de classement et de sélection des variables. Elle est intégrée à la classe des algorithmes GPS introduite par Audet et Dennis (2001) pour les problèmes à variables mixtes continues et de catégorie. Étant donné la valeur erronée de l'objectif à cause du bruit présent dans la fonction, l'algorithme fait appel à la procédure de classement et de sélection dans le but de déterminer le prochain itéré parmi l'ensemble fini de points générés par l'algorithme GPS. Ainsi, l'évaluation de la fonction en un point est donnée par la moyenne de l'échantillon construit par des évaluations répétitives à un même point. D'une part cette procédure permet de déterminer la taille de l'échantillon qui garantit l'obtention d'une bonne estimation de la valeur de l'objectif pour chaque candidat. D'autre part, ce classement permet de sélectionner le point ayant la valeur minimale. Un autre algorithme de recherche directe capable de traiter les fonctions contaminées par un bruit aléatoire contrôlé est celui de Anderson et Ferris (2001), chaque itération débute par la génération d'un nouvel ensemble de points d'essai à partir d'un ensemble de points courant et ceci en appliquant une des opérations *Réflexion*, *Expansion* et *Contraction* de l'algorithme de Nelder et Mead (1965), autour du meilleur point dans l'ensemble des points d'essai. En supposant que plus courte est la distance maximale entre les points générés au cours de l'itération, plus la précision de l'évaluation est meilleure. Le bruit peut alors être contrôlé à chaque évaluation de sorte que son écart type est borné par une quantité qui dépend de la précision de la solution courante. Ceci permet la diminution de l'écart type du bruit à chaque fois que la distance entre les points diminue et par conséquent, augmenter la précision de l'évaluation.

D'autres méthodes pour aborder le bruit existent dans la littérature que nous citerons à la section suivante. Généralement les fonctions obtenues par l'application de ces méthodes sont lisses.

2.4.3 Quelques techniques de lissage

Quand la fonction à optimiser dépend des entrées x et de la variation autour de x, l'optimisation se fait par rapport à la distribution des valeurs sous l'effet de la variation ϵ, c'est-à-dire $f = z(x, \epsilon)$. La méthode décrite par Rhein *et al.* (2014) permet justement de fournir une solution robuste qui minimise z et la variation causée par ϵ. Dans leur travaux, ils visent à caractériser la distribution de probabilité des valeurs de f par la méthode de l'estimation de

densité par noyau. L'estimateur $\hat{f}$ de la densité de f, à partir d'un échantillon $(x_1, ..., x_n)$, est donné par

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\psi_s(x - z(x_i)) = \frac{1}{n}\sum_{i=1}^{n}\psi\left(\frac{x - z(x_i)}{s}\right), \tag{2.19}$$

où ψ est la fonction noyau et s un paramètre de lissage. Ainsi la fonction de répartition de f est donnée par intégration de (2.19).

Une autre technique permettant de construire une fonction lisse à partir d'un ensemble d'observations $(f(x_1), x_1), ..., (f(x_n), x_n)$ où $f : \mathbb{R}^n \to \mathbb{R}$ et $x_i \in \mathbb{R}^n, i = 1, ..., n$ est la méthode de régression par noyau introduite par Nadaraya (1964), elle est décrite ici comme elle apparaît dans Hastie *et al.* (2001). À partir des observations $f(x_i), i = 1, ..., n$ la méthode consiste à calculer l'approximation de la fonction f au point x_0 par une somme pondérée des observations $f(x_i)$. Les poids dépendent de la distance entre le point x_0 et les points dans le voisinage de x_0. La formule de la fonction approchée est donnée par

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{n}\psi_s(x_0, x_i)f(x_i)}{\sum_{i=1}^{n}\psi_s(x_0, x_i)}, \tag{2.20}$$

où la fonction noyau ψ_s est définie par

$$\psi_s(x_0, x) = \mathcal{K}\left(\frac{||x_0 - x_i||}{s}\right), \tag{2.21}$$

avec $\mathcal{K}$ une fonction positive décroissante et s un paramètre de lissage. Il existe plusieurs fonctions de noyau, la fonction Epanechnikov comme celle utilisée dans cet exemple, la fonction Gaussienne, etc.

Li *et al.* (2014) proposent une technique de lissage pour les problèmes convexes et non lisses, $\min_{x \in X} f(x)$ où $f(x) = \mathbb{E}[z(x, \epsilon)]$. Elle consiste à faire une approximation de la fonction par une fonction lisse construite par le produit de convolution de la fonction non lisse par la fonction de densité de la variable aléatoire qui représente le bruit. Le problème devient,

$$\hat{f}(x) = \int_{\mathbb{R}^n} f(x + t)v(t)dt,$$

où ϵ est une variable aléatoire de densité v. Il est garanti que la fonction lissée $\hat{f}(x)$ est differentiable.

Dans le même contexte, Shao *et al.* (1997) illustrent une technique de lissage, sur la fonction énergétique connue sous le nom de « Lennard-Jones », dans un modèle moléculaire qui repose sur la moyenne. Ce problème est considéré comme difficile à résoudre, car le nombre de minima locaux augmente exponentiellement avec le nombre d'atomes. L'idée proposée consiste à remplacer la valeur de la fonction à chaque point par une moyenne pondérée de la fonction énergétique dans le voisinage de ce point. La distribution de probabilité d'une loi gaussienne est utilisée pour déterminer les poids. Le problème de minimisation devient

$$\widetilde{f}_s(x) = \int_{\mathbb{R}^n} H(f(t), s) \exp\left(\frac{-\|x-t\|^2}{\sigma^2}\right) dt,$$

où l'écart-type σ et s constituent les paramètres de lissage et H la fonction permettant de faire la transformation de f en une fonction intégrable. Elle peut être obtenue par la méthode de transformation de systèmes énergétiques Wu (1996) qui repose sur la moyenne des valeurs de f ou bien, la méthode de l'équation de diffusion Kostrowicki *et al.* (1991).

Dans l'optique de se ramener à des fonctions différentiables dans un modèle mathématique, Xavier *et al.* (2014) s'intéressent au problème de localisation dont la modélisation aboutit à un problème de minimisation non différentiable, non convexe et possédant un grand nombre de minima locaux. Les auteurs optent pour une méthode dont la dérivée de premier ordre est requise. Ils se ramènent à un problème équivalent complètement differentiable en appliquant une série de transformations sur la formulation du problème, notamment les contraintes formulées par la fonction distance $\|x-y\|_2$, $(x,y) \in \mathbb{R}^{n\times n}$, qui est remplacée par une fonction $\sqrt{\sum_{l=1}^{n}(x_i-y_i)^2+\gamma^2}$, $\gamma > 0$, dont la dérivée existe.

CHAPITRE 3

UN ALGORITHME DE RECHERCHE DIRECTE ROBUSTE

Ce chapitre décrit la méthode proposée dans ce projet à savoir l'algorithme RobustMads basé sur l'algorithme Mads présenté au chapitre précédent. Cet algorithme vise à réduire l'influence du bruit sur la valeur de l'objectif afin de conduire l'optimisation du problème (1.1) vers un optimum robuste. Cette solution a la particularité de demeurer stable lorsqu'elle est affectée d'une perturbation, c'est-à-dire que la valeur de la fonction objectif varie peu dans le voisinage de la solution optimale.

3.1 Approche algorithmique

L'idée proposée consiste à appliquer une technique de lissage sur les valeurs de la fonction f permettant de diminuer le bruit qui fausse la valeur de l'objectif. En effet, l'approche présentée ici utilise l'algorithme itératif Mads. Et par conséquent, pour déterminer le prochain itéré nous devons comparer les points générés au cours de l'itération courante. Se fier aux évaluations de f n'est pas adéquat à cause du bruit présent dans cette fonction. Par ailleurs, étant donnée la caractéristique de la boîte-noire, on a seulement accès aux valeurs de la fonction. Donc, faire la moyenne en échantillonnant maintes fois au même point x afin de trouver une bonne approximation de la valeur de la fonction démunie de bruit, comme dans les travaux de Sriver *et al.* (2009), est d'une part très couteux, et d'autre part n'éclaire pas sur son allure générale. En effet, la moyenne peut être très affectée par des valeurs extrêmes à cause du bruit. Nous cherchons alors à appliquer une technique de lissage sur la fonction f créant ainsi un algorithme capable de considérer la fonction débruitée.

Une façon de débruiter la valeur de f serait de faire une somme continue pondérée sur l'ensemble $\mathbb{R}^n$, idéalement on escompte minimiser la fonction débruitée $\widetilde{f}(x)$ définie par

$$\widetilde{f}(x) \;=\; \int_{u\in\mathbb{R}^n} w(u,x)f(u)du, \tag{3.1}$$

où w est un poids attribué à chaque valeur $f(x)$ tel que

$$\int_{u\in\mathbb{R}^n} w(u,x)du = 1. \tag{3.2}$$

Étant donnée la structure complexe de la fonction f, il serait alors impossible de calculer

cette intégrale car nous avons seulement accès aux valeurs de la fonction mais pas à son expression analytique puisqu'il s'agit d'une boîte-noire. Une façon plus adéquate pour réduire le bruit dans la fonction f est d'appliquer le lissage sur la fonction f en construisant une fonction $\widetilde{f}$ définie par une somme de toutes les évaluations des itérations précédentes. Pour ce faire, on a besoin de garder en mémoire toutes ces valeurs.

3.1.1 Pilotage de MADS

L'utilisation classique de l'algorithme MADS permet l'évaluation de la fonction objectif tant que le budget maximal n'est pas atteint ou qu'un autre critère d'arrêt a été activé. Dans ce projet, l'utilisation de MADS se fait par itération, c'est-à-dire que l'algorithme s'arrête au bout d'une seule itération qui inclut l'évaluation du point de départ ainsi qu'un bloc de points générés par MADS. Ceci permet à l'utilisateur de gérer par lui-même le choix du point de départ de la prochaine itération et de définir à quel moment une itération est considérée comme un succès.

À la figure 3.1 est donné un schéma de l'évaluation par bloc de points W_k pour l'itération k. La boîte-noire représentée retourne à chaque itération un bloc d'évaluations $f(x_i)_{i=1,2,...I_k}$ qui correspondent aux entrées $x_i \in W_k, i = 1, 2, ...I_k$ où $I_k =\mid W_k \mid$. Ces valeurs correspondent aux valeurs de la fonction objectif contaminée par le bruit qui peut être présent sur les entrées $x_i, i = 1, 2, ...I_k$, ou la sortie f.

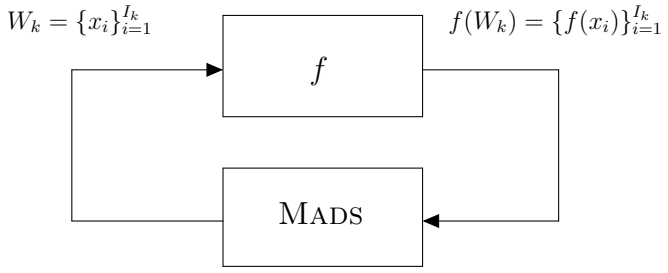


Figure 3.1 Schéma de l'évaluation de la boîte-noire par blocs.

3.1.2 Construction de la fonction débruitée

À chaque itération, la fonction f est évaluée à un ensemble fini de points générés lors de la recherche globale et la sonde locale de l'algorithme MADS. Ces points constituent le

bloc W_k. Soit V_k l'ensemble des points où f a déjà été évaluée au début de l'itération k. La k-ème itération débute par l'évaluation de f dans le bloc de points W_k ce qui nous permet de stocker les nouvelles informations $\{f(x_i)\}_{i=1}^{I_k}$.

Soit, $\widetilde{f_k}(x_i)$ la fonction débruitée de la fonction f calculée pour chaque point $x \in W_k$ par une pondération de toutes les évaluations de f. C'est-à-dire, les évaluations dans W_k ainsi que toutes les évaluations depuis la première itération qui sont stockées dans l'ensemble V_k. L'utilisation de toutes les évaluations disponibles permet d'avoir davantage d'informations sur la fonction f, et par conséquent, de construire une fonction débruitée représentative de f. Ainsi, chaque valeur de f au point $x \in W_k$ est remplacée par la valeur de $\widetilde{f_k}$ associée à ce point ainsi qu'à l'itération courante. La fonction débruitée est calculée pour chaque point de W_k comme suit :

$$\widetilde{f_k}(x) \ := \ \frac{1}{P_k(x)} \sum_{u \in V_k \cup W_k} w(u,x) f(u) \tag{3.3}$$

où

$$P_k(x) \ = \ \sum_{u \in V_k \cup W_k} w(u,x) \tag{3.4}$$

est la somme des poids qui permet la normalisation selon (3.2).

Définition des poids

La motivation est de pouvoir attribuer un poids, $w(u,x)$, proche de zéro aux valeurs de f dont les points correspondants sont éloignés du point x, et un poids proche de un aux valeurs dont les points sont proches de x. Pour définir le poids associé à chaque valeur, on a utilisé la densité de probabilité d'une loi normale, dont la motivation est soutenue par sa forme en cloche qui permet de l'utiliser comme mesure de rapprochement et d'éloignement par rapport à un point donné. Effectivement, soit Y la variable aléatoire gaussienne représentant la distance euclidienne entre les points d'essai de moyenne $\mu = 0$ et de variance σ^2, dont la densité de probabilité est donnée par :

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}}, y \in \mathbb{R}. \tag{3.5}$$

Plus les points se trouvent à une distance proche du point considéré, plus les valeurs correspondantes ont un poids supérieur, i.e., $g(y)$ avec y proche de zéro. Ainsi, le poids attribué à

la valeur correspondant à un point $u \in V_k \cup W_k$ est donné par la fonction composée d'une densité de probabilité et la fonction distance. Plus tard, à la section 4.3 l'algorithme sera testé pour plusieurs valeurs du paramètre σ^2.

Les équations (3.3) et (3.4) deviennent alors,

$$\widetilde{f_k}(x) \;=\; \frac{1}{P_k(x)} \sum_{u \in V_k \cup W_k} g \circ d(x,u) f(u) \tag{3.6}$$

$$P_k(x) \;=\; \sum_{u \in V_k \cup W_k} g \circ d(x,u). \tag{3.7}$$

où, $d(x,u)$ est la distance euclidienne entre le point x et u.

Mise à jour de la fonction débruitée pour les points V_k

L'utilisation d'un historique des évaluations V_k est nécessaire pour la construction de la fonction débruitée. Néanmoins, il faut mettre à jour cet ensemble à chaque fois que de nouveaux points sont générés. Ce qui conduit à une meilleure qualité de lissage car la fonction débruitée est recalculée pour chaque point de V_k avant la fin de chaque itération.

La figure (3.2) résume les étapes principales de l'algorithme RobustMads. À l'itération k, Mads analyse l'historique des points précédemment évalués et génère un bloc W_k de candidats afin d'améliorer la solution courante. Ensuite, la fonction bruitée f est évaluée à tous ces points et la fonction débruitée $\widetilde{f}_k$ est construite. Les valeurs de $\widetilde{f}_k$ à tous les points de V_k et de W_k sont ensuite retournées à Mads. Ces valeurs permettront à Mads de mettre ses paramètres internes à jour, de déterminer si l'itération est un succès ou non et ensuite de passer à l'itération suivante. Cette utilisation de l'algorithme RobustMads est similaire à l'utilisation de Mads représentée à la figure (3.1), dans les deux cas une liste d'évaluations est passée à l'algorithme, la seule difference est que ici Mads est piloté itération par itération, ce qui lui permet à chaque itération de chercher un meilleur itéré dans l'historique V_k.

Une fois le lissage des valeurs de W_k effectué, les valeurs sont stockées puis envoyées à Mads et avant la fin de l'itération, la fonction $\widetilde{f_k}$ est mise à jour pour les points V_k en considérant les nouveaux points de W_k. La proposition suivante permet justement de donner la formule qui vise à obtenir les valeurs de la fonction $\widetilde{f}$ aux points de V_k. Ainsi $\widetilde{f}$ s'obtient directement par l'utilisation des ses valeurs précédentes.

Proposition 8. *Pour chaque point $x \in V_k$ la fonction $\widetilde{f_k}(x)$ est mise à jour de la façon*

suivante :

$$\widetilde{f_k}(x) \;:=\; \frac{1}{P_k(x)}\left(P_{k-1}(x)\widetilde{f}_{k-1}(x) + \sum_{u\in W_k} g\circ d(x,u)f(u)\right) \tag{3.8}$$

avec

$$P_k(x) \;=\; P_{k-1}(x) + \sum_{u\in W_k} g\circ d(x,u). \tag{3.9}$$

Preuve. À l'itération k la fonction $\widetilde{f}_k(x)$ a été calculée pour chaque point de W_k en considérant les points appartenant aux ensembles W_k et V_k selon (3.6). En revanche, les points de V_k ont été calculés seulement en prenant en compte les points V_k à l'itération précédente. Pour mettre à jour la fonction $\widetilde{f}_k(x)$ en ces points, on doit la calculer en considérant aussi les points de W_k.

Pour chaque point $x \in V_k$

$$\begin{aligned}
\widetilde{f_k}(x) \;&:=\; \frac{1}{P_k(x)} \sum_{u\in W_k\cup V_k} g\circ d(x,u)f(u) \\
&=\; \frac{1}{P_k(x)}\left(\sum_{u\in V_k} g\circ d(x,u)f(u) + \sum_{u\in W_k} g\circ d(x,u)f(u)\right) \\
&=\; \frac{1}{P_k(x)}\left(\sum_{u\in V_{k-1}\cup W_{k-1}} g\circ d(x,u)f(u) + \sum_{u\in W_k} g\circ d(x,u)f(u)\right)
\end{aligned}$$

à l'itération $k-1$ la fonction $\widetilde{f}_{k-1}(x)$ a été calculée selon (3.6). En remplaçant par sa valeur dans l'égalité précédente on obtient

$$=\; \frac{1}{P_k(x)}\left(P_{k-1}(x)\widetilde{f}_{k-1}(x) + \sum_{u\in W_k} g\circ d(x,u)f(u)\right).$$

Et $P_k(x)$, s'obtient simplement en ajoutant la somme des poids correspondants aux points de W_k par rapport à x au poids de l'itération précédente.

$$P_k(x) = P_{k-1}(x) + \sum_{u\in W_k} g\circ d(x,u)f(u)$$

□

Ainsi, avant la fin de chaque itération, l'ensemble W_k est ajouté à l'historique $V_{k+1} \leftarrow V_k \cup W_k$.

La fonction débruitée, $\widetilde{f_k}$, est construite d'une façon dynamique car sa valeur diffère au cours des itérations. De plus, à chaque fois qu'un nouveau bloc de points est à considérer, tous les points sont mis à jour ce qui permet de l'affiner et de la recalibrer.

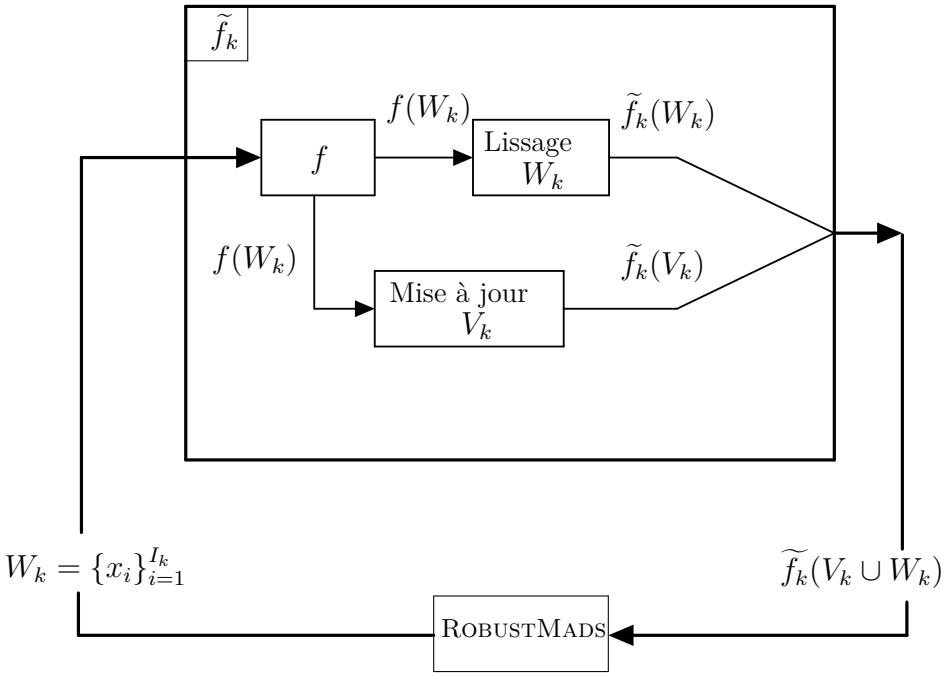


Figure 3.2 L'itération k de RobustMads.

Dans un contexte où les fonctions traitées sont des boîtes-noires, leur évaluation est difficile et très coûteuse en temps d'exécution. Cette façon de mettre à jour la fonction n'est pas couteuse car elle consiste à faire une somme pondérée des valeurs qui sont déjà stockées en mémoire. En pratique, dans l'équation (3.8), la valeur $\widetilde{f}_{k-1}(x)$ et le poids correspondant $P_k(x)$ sont stockés dans l'historique V_k. De plus, la somme est effectuée sur l'ensemble W_k où f a déjà été évaluée.

À la figure 3.3 sont présentées les principales étapes de l'algorithme RobustMads.

RobustMads pour les fonctions bruitées

- Initialisation :
  - Soit x_0 un point de départ
  - Évaluer f au point x_0
  - Stocker $(x_0, f(x_0))$ dans la cache : $V_0 \leftarrow \{x_0, f(x_0)\}$
  - La fonction débruitée au point x_0 : $\widetilde{f_0}(x_0) \leftarrow f(x_0)$
  - Initialiser le compteur des itérations : $k \leftarrow 0$
- Recherche globale et locale :
  - Effectuer la search et la poll sur l'ensemble W_k, tel que $W_k \cap V_k = \emptyset$.
- $\forall x \in W_k$: Calculer $\widetilde{f_k}(x)$ et $P_k(x)$ selon (3.6) et (3.7)
- $\forall x \in V_k$: $\widetilde{f}(x) \leftarrow \widetilde{f_k}(x)$ où $\widetilde{f_k}(x)$ est calculée selon (3.8)
- Mises à jour :
  - $V_{k+1} \leftarrow V_k \cup W_k$
  - Choisir $x_{k+1} \in \text{argmin}\{\widetilde{f_k}(x) : x \in V_{k+1}\}$.
  - Si $x_{k+1} \in W_k$ alors l'itération est un succès : $\Delta^m_{k+1} \geq \Delta^m_k$.
    Sinon, si x_{k+1} n'est pas un centre de sonde, alors $\Delta^m_{k+1} = \Delta^m_k$.
    Sinon, l'itération est un échec, alors $\Delta^m_{k+1} < \Delta^m_k$.
  - Incrémenter $k \leftarrow k+1$ et aller à search et poll.

Figure 3.3 L'algorithme RobustMads.

3.2 Description des étapes de l'algorithme RobustMads

L'algorithme Mads présenté à la section 2.2.3 définit une itération comme étant un succès dès qu'un nouveau point, parmi le bloc W_k, est meilleur que l'itéré courant. Lors de l'application originalement prévu de cette algorithme, il est certain qu'il ne peut y avoir un meilleur point dans l'historique. Par contre, dans ce projet, les évaluations précédentes sont mises à jour à chaque itération k. Ce qui incite à faire le choix du prochain itéré x_{k+1} en comparant la valeur de $\widetilde{f_k}$ au point courant aux valeurs de la fonction débruitée aux points de W_k et de V_k. De ce fait, l'itéré prochain est donné par : $x_{k+1} \in \text{argmin}\{\widetilde{f_k}(x) : x \in V_{k+1}\}$ où $V_{k+1} = V_k \cup W_k$ et on distingue trois cas :

- L'itération est un succès si le point x_{k+1} est un point qui a été généré au cours de l'itération courante, i.e., $x_{k+1} \in W_k$. Un nouveau point a été trouvé alors la taille du treillis est augmentée : $\Delta^m_{k+1} \leftarrow 4\Delta^m_k$;
- Un autre cas est quand le point $x_{k+1} \in V_k$ et x_{k+1} n'est pas un centre de sonde, alors c'est un point dont la valeur de $\widetilde{f_k}$ a été mise à jour, i.e., à l'itération $k-1$, $\widetilde{f}_{k-1}(x_{k+1}) > \widetilde{f}_{k-1}(x_k)$ et $\widetilde{f}_{k-1}(x_{k+1}) > \widetilde{f}_k(x_{k+1})$. Dans ce cas la taille du treillis reste

invariante : $\Delta^m_{k+1} \leftarrow \Delta^m_k$;

— L'itération est un échec si le point x_{k+1} a été généré au cours des itérations précédentes $x_{k+1} \in V_k$ et est le centre de sonde de l'itération k. Comme x_{k+1} est un minimum local la taille du treillis est réduite : $\Delta^m_{k+1} \leftarrow 1/4\Delta^m_k$.

3.3 Analyse de convergence de RobustMads

Les résultats de l'analyse de convergence pour RobustMads sont très proches des résultats de premier ordre de Mads. Ils reposent sur l'hypothèse que les itérés générés par RobustMads appartiennent à un ensemble borné. Ainsi découle le théorème suivant.

Théorème 9. *Si tous les points générés par* RobustMads *appartiennent à un ensemble borné, alors* $\liminf_{k\to\infty} \Delta^m_k = 0$.

Preuve. En fait, tel qu'indiqué dans Audet et Dennis (2006) la preuve donnée par Audet et Dennis (2003) ne fait pas intervenir l'objectif. Elle s'applique donc ici. □

Rappelons qu'il existe un lien entre les deux paramètres Δ^m_k et Δ^p_k, voir (Proposition 2).

Définition 10. *Dans le cas sans contraintes, la sous-suite des centres de sonde* $\{x_k\}_{k\in K}$*, dont les itérations ont été qualifiées d'échec, est dite une sous-suite raffinante si* $\{\Delta^p_k\}_{k\in K}$ *converge vers zero, où* K *est un sous-ensemble d'indices.*

Le théorème suivant découle de l'article de Audet et Dennis (2003), il assure l'existence d'une sous-suite raffinante convergente.

Théorème 11. *Sous l'hypothèse que les itérés appartiennent à un ensemble borné, Il existe une sous-suite raffinante convergente.*

Preuve. Soit K l'ensemble des indices des itérés dont les itérations ont été qualifiées d'échec. Puisque la taille du treillis est réduite à chaque échec, le théorème 9 montre que pour un K suffisamment grand $\{\Delta^m_k\}_{k\in K} \to 0$, et étant donné que les itérés sont dans un ensemble borné, alors on peut en extraire une sous-suite convergente. □

On énonce le théorème suivant qui est une réécriture de la définition 10.

Théorème 12. *Si* $\hat{x}$ *est la limite d'une sous-suite raffinante, alors* $\hat{x}$ *est la limite d'optima locaux de* $\tilde{f}_k$ *sur des treillis devenant infiniment fins.*

Preuve. Soit $\{x_k\}_{k\in K}$ la sous suite d'optima locaux de la fonction $\tilde{f}_k$. Il découle de la définition 10 que pour un K suffisamment grand, la taille du treillis tend vers zéro. Et d'après le théorème 11, la sous suite converge vers $\hat{x}$. □

Les résultats proposés ici s'apparentent aux résultats fondamentaux de MADS ou l'objectif ne varie pas à chaque itération. Ce sont les résultats de base. Les corollaires plus forts dans MADS n'ont pas leur contrepartie ici car ils requièrent plus d'hypothèses sur l'objectif comme la différentiabilité stricte. Dans le cas où la fonction à optimiser est bruitée les hypothèses émises pour la convergence de MADS ne peuvent pas être énoncées ici car la fonction utilisée est construite dynamiquement. À chaque itération k l'optimisation est lancée sur une nouvelle fonction $\tilde{f}_k$.

3.4 Implémentation de l'algorithme

Dans cette section sont expliqués les principaux paramètres de NOMAD pour l'utilisation de MADS dans le processus de l'optimisation.

Stratégie de recherche

La stratégie de recherche utilisée dans ce projet est la recherche spéculative fixée par défaut, telle qu'évoquée dans la section 2.3. Par contre, la sonde locale n'est pas opportuniste, c'est-à-dire que même si un point x meilleur que le centre de sonde actuel x_k est trouvé, l'itération courante k ne prend fin qu'après l'évaluation de tous les points dans la liste P_k (2.7). La nouvelle itération $k+1$ commence avec pour centre de sonde le meilleur point parmi les points évalués depuis le début de l'itération.

L'évaluation par blocs

Dans le cas de ce projet, le paramètre représentant la taille des blocs est fixé à n, où n est le nombre de variables pour le problème traité. L'évaluation de chaque bloc se fait suivant la stratégie opportuniste décrite plus haut. Ainsi, l'algorithme va évaluer la fonction objectif dans l'ensemble constitué de l'union des points générés lors de la recherche globale et la sonde locale d'une itération de l'algorithme MADS, puis un bloc d'évaluations est envoyé à MADS comme illustré à la figure 3.1.

L'historique des évaluations

Dans NOMAD, il existe une structure de données, nommée *cache*, qui permet le stockage de tous les points où f a été évaluée depuis le début de l'exécution de l'algorithme. Son avantage est d'empêcher la réévaluation d'un point déjà visité qui peut être très couteuse. L'utilisation d'une cache est nécessaire pour la construction de la fonction lisse. En effet, le besoin de son utilisation réside dans le fait que, d'une part, tous ses points sont utilisés ce

qui conduit à une meilleure qualité de lissage, et d'autre part, l'itéré courant est comparé aux points générés à l'itération courante et aussi aux points de la cache. Ceci permet de tenter de trouver un meilleur point dans la cache, étant donné que toutes les valeurs correspondant à ses points ont été recalculées. Grâce au pilotage de MADS par itération de la section 3.1.1, l'historique des évaluations peut être géré de l'extérieur de MADS et ainsi permettre la recherche d'un meilleur point.

Le critère d'arrêt

Pour le critère d'arrêt on se base sur une valeur minimale de la taille du treillis Δ_k^m, c'est-à-dire que lorsque la taille de treillis glisse sous un certain seuil, l'algorithme s'arrête. Pour des fins de comparaisons, un second critère consiste en un budget d'évaluations.

CHAPITRE 4

RÉSULTATS NUMÉRIQUES ET DISCUSSIONS

Le présent chapitre décrit les résultats obtenus dans ce projet de recherche et il est construit de la façon suivante. La section 4.1 est consacrée à la définition des familles de problèmes et la démarche à suivre pour la construction d'un ensemble de fonctions tests. Ces fonctions, qui constituent nos boîtes-noires, sont conçues artificiellement à partir de fonctions tests non-lisses issues de la littérature, auxquelles deux types de bruit ont été ajoutés afin de réaliser une batterie de tests. Ensuite, la section 4.2 présente les mesures de robustesse utilisées. Puis, les résultats numériques de l'application de l'algorithme proposé au chapitre 3 sont exposés à la section 4.3. La robustesse des solutions obtenues seront comparées à la robustesse des solutions obtenues avec MADS. Pour finir, une synthèse des résultats est donné à la section 4.4.

4.1 Deux familles de problème tests

On s'intéresse dans ce mémoire à étudier deux familles de problèmes, les problèmes de type stochastique et les problèmes de type déterministe. Les deux familles de problèmes tests ont été créées à partir du benchmark proposé par Moré et Wild (2009). On y trouve quatre types de problèmes : les problèmes lisses, les problèmes non différentiables, les problèmes déterministes et les problèmes bruités par un bruit stochastique. Le choix s'est porté sur les problèmes stochastiques et sans contraintes définis sous la forme :

$$\min_{x\in\mathbb{R}^n} z(x) = \sum_{i=1}^{q} z_i(x)^2. \tag{4.1}$$

où z est la somme des carrés de q fonctions bruitées.

Nous allons expliquer comment, à partir des problèmes tests déjà bruités de type (4.1), les deux familles de problèmes tests à traiter ici ont été construites. Comme il a été expliqué au chapitre 2, l'incertitude peut être présente à la fois sur les variables du modèle et sur la valeur de la solution. Ces deux cas d'incertitude sont traités dans ce mémoire. À des fins de clarté, les deux fonctions bruitées pour les problèmes stochastiques et déterministes sont notées par f, nous indiquerons à chaque fois de quel type de problème il s'agit. La figure (4.1) montre la façon dont la fonction bruitée, f, est construite. À partir de la boîte-noire z, une perturbation sur les entrées $x \in W$ est ajoutée, en plus d'un bruit à la sortie.

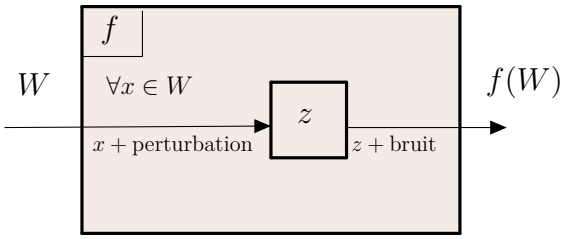


Figure 4.1 La construction artificielle de la fonction bruitée f.

4.1.1 La familles des problèmes stochastiques

Un problème stochastique est défini ici tel que présenté par Larson et Billups (2013) et Larson et Wild (2013). À partir d'une fonction de base, z, il est ajouté un bruit défini par une variable aléatoire ϵ_x, indépendante et identiquement distribuée (iid), de moyenne nulle et de variance finie σ^2_{bruit} :

$$\min_{x\in\mathbb{R}^n} \quad f(x) = z(x)(1+\epsilon_x). \tag{4.2}$$

De plus, nous ajoutons une perturbation υ_x autour du point courant x, le problème (4.2) devient alors :

$$\min_{x\in\mathbb{R}^n} \quad f(x) = z(x(1+\upsilon_x))(1+\epsilon_x), \tag{4.3}$$

où υ_x est la réalisation d'une variable aléatoire suivant la même loi de probabilité que la variable ϵ_x. Cette façon de bruiter les entrées est similaire à celle de Anderson et Ferris (2001). Rappelons que par construction, les fonctions stochastiques que nous traitons ici ne sont pas telles que deux évaluations en un point x donnent deux résultats différents. Dans notre cas le bruit ϵ_x et la perturbation υ_x dépendent du point x, i.e., c'est la même perturbation et le même bruit qui sont à chaque fois affectés au même point x.

4.1.2 La familles des problèmes déterministes

On peut trouver, dans les travaux de Moré et Wild (2009), qu'un problème déterministe est caractérisé par la présence d'une composante déterministe désignant le bruit, soit

$$f(x) = (1+\epsilon_{bruit}\phi(x))z(x), \tag{4.4}$$

où ϵ_{bruit} est un scalaire représentant le niveau de bruit et $\phi : \mathbb{R}^n \to [-1; 1]$ est la fonction bruitée définie par le polynôme de Chebyshev T_3 telle que présenté à la section 2.4.2. Les entrées sont eux aussi perturbées de façon déterministe, le problème (4.4) s'écrit comme

$$f(x) = (1 + \epsilon_{bruit}\phi(x))z(x(1 + \epsilon_{bruit}\phi(x))). \tag{4.5}$$

4.2 Les mesures de robustesse pour une solution optimale

Une fois les fonctions tests définies, nous pouvons présenter les résultats de l'optimisation, mais avant d'aller plus loin il convient de bien définir comment qualifier une solution de robuste. En effet, il existe plusieurs façons pour quantifier la robustesse d'une solution, mais la définition d'une mesure dépend étroitement de l'interprétation donnée au terme robuste et du type de problème traité. Dans certains cas, une solution robuste est définie comme étant la solution qui demeure inchangée lorsque des paramètres dans le modèle changent. Dans d'autres cas, elle est définie par la solution obtenue par un nombre d'évaluations restreint. Dans ce mémoire, une solution robuste est telle que la fonction reste stable dans le voisinage de cette solution. Une illustration de la solution espérée est donnée à la figure (1.3).

La mesure de la robustesse fait naturellement appel à une quantification de la dispersion de la valeur de l'objectif autour de la solution optimale. Ce que nous présentons comme résultats sont une analyse des solutions obtenues sur les problèmes bruités construits artificiellement et la robustesse est mesurée à la fin de l'optimisation. Bien entendu, et sans perdre de généralité, la robustesse n'est pas le seul critère à prendre en compte, nous cherchons une solution qui donne un bon compromis entre robustesse et optimalité.

Dans un premier lieu, l'optimisation est lancée sur les problèmes de type (4.3), puis les solutions obtenues sont analysées. On s'intéresse au comportement de la fonction f autour de la solution optimale. Les mesures proposées ici reposent sur l'évaluation de la fonction au voisinage de la solution optimale, soit l'échantillon aléatoire $\mathcal{N}$ de taille finie généré par hypercube latin (LHS *Latin Hypercube Sampling*), de McKay *et al.* (1979), autour de la solution optimale. Cet ensemble représente le voisinage d'une solution dans un rayon $r \in \mathbb{R}$, choisi relativement à cette solution. Trois mesures sont présentées dans ce mémoire.

4.2.1 La mesure du pire cas

Pour cette mesure $m_1(x^*)$, nous considérons l'échantillon aléatoire $\mathcal{N}$ dans un voisinage de x^* et on s'intéresse à la valeur maximale de f. Étant donné qu'on se trouve dans un contexte de minimisation, la valeur maximale nous donne la pire valeur autour de la solution optimale. La mesure introduite ici vise à identifier l'algorithme ayant la plus petite pire valeur

de l'objectif.

$$m_1(x^*) \ = \max_{x \in \mathcal{N}} \ \ f(x). \tag{4.6}$$

4.2.2 La mesure basée sur la variance

La mesure utilisée ici a été proposée par Wang et Zou (2010) et elle repose sur le calcul d'un ratio qui mesure la variation locale de la fonction f par rapport à la variation locale de chaque variable. Dans notre cas, on s'intéresse seulement à mesurer la solution optimale. Ainsi, au lieu de calculer ce ratio pour tous les points, il est calculé pour chaque solution finale produite par les algorithmes à la fin de l'optimisation. Pour cette mesure, nommée $m_2(x^*)$, nous considérons à nouveau l'échantillon aléatoire $\mathcal{N}$ autour de x^*. Soit $\bar{f}$ la moyenne échantillonnale de f autour de la solution optimale,

$$\bar{f} \ = \ \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} f(x), \tag{4.7}$$

et soit S_f^2 la variance de cet échantillon définie par

$$S_f^2 \ = \ \frac{1}{|\mathcal{N}| - 1} \sum_{x \in \mathcal{N}} (f(x) - \bar{f})^2. \tag{4.8}$$

Soit $\bar{x} = (\bar{x}_1, ..., \bar{x}_j, ..., \bar{x}_n)$, le vecteur dont chaque coordonnée $\bar{x}_j, j = 1, ..., n$ est donnée par la moyenne des j-èmes coordonnées des points $x = (x_1, ..., x_j, ...x_n) \in \mathcal{N}$ comme :

$$\bar{x}_j \ = \ \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} x_j, \ \forall j = 1, ..., n \tag{4.9}$$

et soit S_j^2 la variance correspondant à chaque coordonnée $\bar{x}_j, j = 1, ..., n$ tel que :

$$S_j^2 \ = \ \frac{1}{|\mathcal{N}| - 1} \sum_{x \in \mathcal{N}} (x_j - \bar{x}_j)^2, \ \forall j = 1, ..., n. \tag{4.10}$$

La variation locale de x^* est définie par :

$$S_{x^*} \ = \ \frac{1}{n} \sum_{j=1}^{n} S_j, \tag{4.11}$$

et la mesure de robustesse est donnée par le ratio suivant :

$$m_2(x^*) \;=\; \frac{S_f}{S_{x^*}}. \tag{4.12}$$

Dans ce cas, m_2 mesure la variation de la fonction par rapport à la variation de la solution. Il est souhaitable d'avoir une solution qui, lorsqu'elle est elle-même perturbée, produit des variations légères sur la fonction. Alors, plus ce ratio est bas plus la solution est robuste.

4.2.3 L'écart type

Cette mesure est une mesure classique pour mesurer la dispersion d'un échantillon aléatoire. Elle est donnée par :

$$m_3(x^*) \;= \sqrt{\frac{1}{|\mathcal{N}|-1}\sum_{x\in\mathcal{N}}(f(x)-\bar{f})^2}, \tag{4.13}$$

où $\bar{f}$ est la moyenne des évaluations dans l'ensemble $\mathcal{N}$ donnée par l'équation (4.7). Et comme pour les deux autres mesures, une petite valeur de cette mesure, m_3 indique la robustesse de la solution.

4.2.4 La mesure basée sur la moyenne

Cette mesure représente la valeur moyenne de la fonction dans le voisinage de la solution x^*. Elle est donnée par l'équation (4.6) :

$$m_4(x^*) \;= \bar{f} \;= \frac{1}{|\mathcal{N}|}\sum_{x\in\mathcal{N}} f(x).$$

Comme pour toutes les autres mesures, il est souhaitable d'avoir une petite valeur de $m_4(x^*)$.

4.3 Résultats numériques

Pour la présentation des résultats, l'algorithme RobustMads a été testé sur un ensemble de problèmes $\mathcal{P}$, composé de 111 problèmes tests, en faisant varier le paramètre de lissage, σ^2 défini à la section 3.1.2, pour trois valeurs différentes, soit $\sigma^2 \in \{0,01; 0,1; 0,2\}$. Les résultats retournés par ces trois algorithmes sont comparés à ceux retournés par l'algorithme Mads.

Afin de comparer la robustesse de la solution retournée par chaque algorithme, notre analyse repose sur trois mesures de robustesse introduites à la section précédente. L'outil présenté ici est analogue aux profils de performance introduits par Moré et Wild (2009), et

permet d'identifier clairement le meilleur algorithme. Celui retournant une solution robuste dans la plupart des problèmes tests. On considère l'ensemble des problèmes $\mathcal{P}$ et un ensemble d'algorithmes $\mathcal{A}$. La robustesse d'une solution optimale retournée par l'algorithme $a \in \mathcal{A}$ pour le problème $p \in \mathcal{P}$ est mesurée par $m_i, \forall i \in \{1, 2, 3\}$. Soit $t^i_{p,a}$, la robustesse calculée en utilisant la mesure i pour le problème $p \in \mathcal{P}$ en appliquant l'algorithme $a \in A$, et $r^i_{p,a}$ le ratio de robustesse. Ce ratio interprète, pour une mesure fixée, l'écart de la valeur de la mesure pour un problème p par rapport à la meilleure valeur de cette mesure pour tous les algorithmes. Ce ratio est donné en pourcentage et est défini par

$$r^i_{p,a} \;=\; \frac{t^i_{p,a} - \min\{t^i_{p,a} : a \in \mathcal{A}\}}{|\min\{t^i_{p,a} : a \in \mathcal{A}\}|},\ i = 1, 2, 3. \tag{4.14}$$

La robustesse d'un algorithme a est donnée par le profil de robustesse :

$$\rho^i_{p,a}(\alpha) \;=\; \frac{1}{|\mathcal{P}|}|\{p \in \mathcal{P} : r^i_{p,a} * 100 \leq \alpha\}|,\ i = 1, 2, 3. \tag{4.15}$$

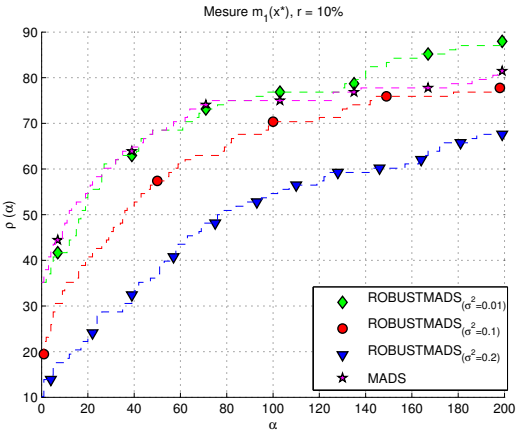
où $\rho^i_{p,a}$ est la proportion des problèmes qui ont été résolus pour une tolérance au plus égale à $\alpha\%$. L'algorithme qui résout le plus de problèmes avec une faible tolérance est le meilleur algorithme, car la solution qu'il retourne est celle qui donne le plus souvent une faible valeur de la mesure m_i et, par conséquent, la solution la plus robuste. Dans ce qui suit les résultats sont présentés sous forme de profils de performance et dans chacune des figures on y trouve plusieurs courbes, chacune représente un algorithme. L'axe des abscisses représente la tolérance $\alpha(\%)$ exprimée en pourcentage et l'axe des ordonnées représente la proportion des problèmes résolus pour une tolérance $\alpha(\%)$ fixée. On s'intéresse aux algorithmes qui résolvent le plus de problèmes avec une faible tolérance.

4.3.1 Résultats pour les problèmes stochastiques

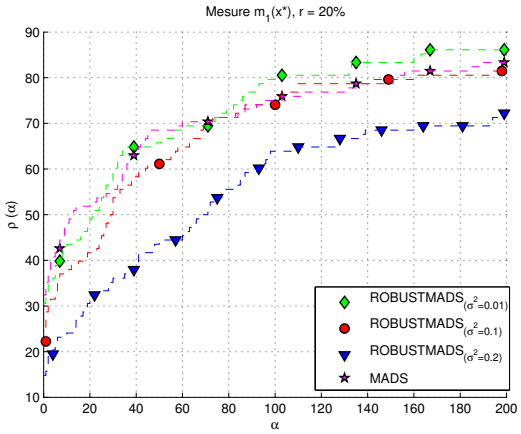
Dans cette sous-section sont présentés les résultats de l'optimisation, des problèmes stochastiques (4.3), par l'algorithme RobustMads et en considérant les trois mesures de robustesse présentées auparavant. Les tests ont été effectués pour trois niveaux de bruit $\sigma^2_{bruit} \in \{0,01; 0,2; 1\}$.

La mesure du pire cas

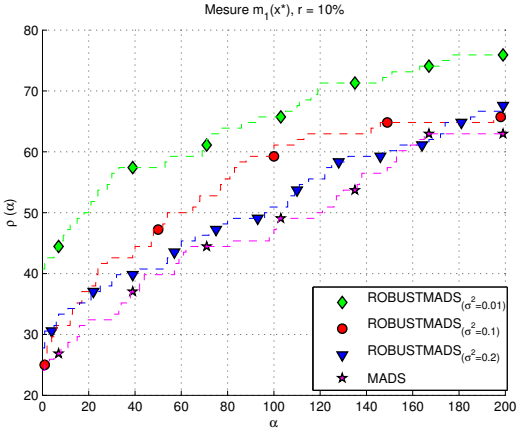
Ces résultats ont été obtenus en utilisant la mesure du pire cas. À la figure (4.2) six graphiques sont représentés. Pour les graphiques (4.2a) et (4.2b) le niveau de bruit est faible ($\sigma^2_{bruit} = 0,01$), pour les deux autres graphiques (4.2c) et (4.2d) le bruit est de niveau plus



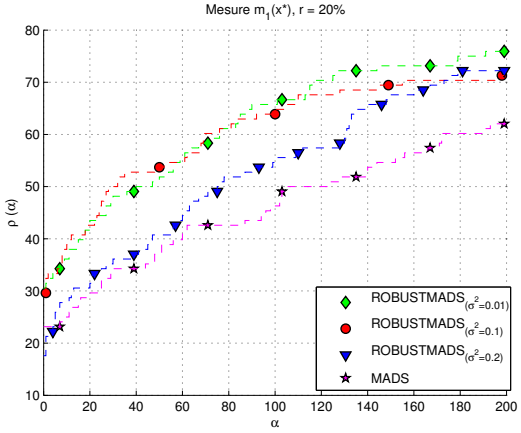
(a) $\sigma^2_{bruit} = 0,01$



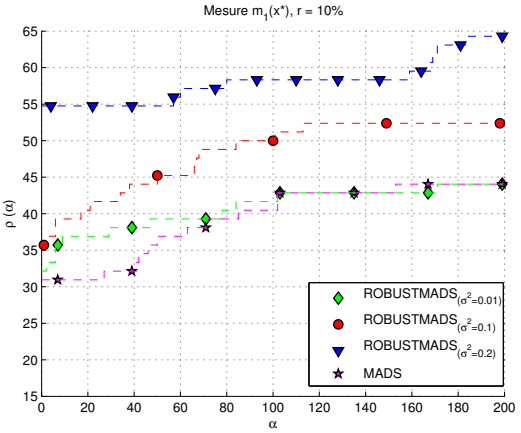
(b) $\sigma^2_{bruit} = 0,01$



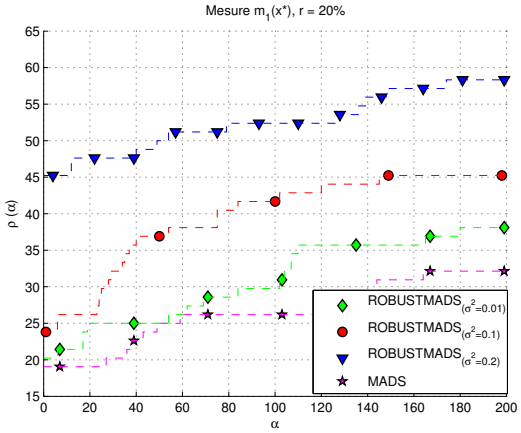
(c) $\sigma^2_{bruit} = 0,2$



(d) $\sigma^2_{bruit} = 0,2$



(e) $\sigma^2_{bruit} = 1$



(f) $\sigma^2_{bruit} = 1$

Figure 4.2 Profils de robustesse pour la mesure du pire cas. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\sigma^2_{bruit} \in \{0,01; 0,2; 1\}$ et la tolérance α exprimée en pourcentage.

fort ($\sigma^2_{bruit} = 0,2$) et pour les derniers (4.2e) et (4.2f) le bruit est le plus élevé ($\sigma^2_{bruit} = 1$). Dans toutes les figures l'axe des abscisses représente la tolérance α. Les profils sont tracés dans l'intervalle allant jusqu'à $\alpha = 200\%$ pour des fins de clarté.

Les graphiques (4.2a) et (4.2b) se distinguent par le rayon du voisinage de la solution optimale. Pour le graphique (4.2a) le rayon est $r = 10\%$ et pour (4.2b) $r = 20\%$. Les graphiques montrent que les trois algorithmes qui sont, MADS, ROBUSTMADS$_{(\sigma^2=0,01)}$ et ROBUSTMADS$_{(\sigma^2=0,1)}$, résolvent plus de 80% des problèmes pour les deux rayons. Pour un voisinage proche de la solution, MADS et ROBUSTMADS$_{(\sigma^2=0,01)}$ ont des comportements semblables et dominent les autres algorithmes. Cependant pour une tolérance de $\alpha < 20\%$, MADS est meilleur et il résout plus que 50% des problèmes. Il est rattrapé par ROBUSTMADS$_{(\sigma^2=0,01)}$ à partir de $\alpha > 20\%$. Pour un rayon plus large, ROBUSTMADS$_{(\sigma^2=0,1)}$ rejoint MADS et ROBUSTMADS$_{(\sigma^2=0,01)}$ et pour $\alpha \in [24\%, 40\%]$, ROBUSTMADS$_{(\sigma^2=0,01)}$ est meilleur. Quant à l'algorithme ROBUST- MADS$_{(\sigma^2=0,2)}$, c'est l'algorithme le moins performant pour les deux rayons. À $\alpha = 20\%$, il résout 20% des problèmes alors que ROBUSTMADS$_{(\sigma^2=0,1)}$ résout le double et les autres environs 55% des problèmes dans un rayon de 10%. Par contre, si on examine les deux algorithmes ROBUSTMADS$_{(\sigma^2=0,2)}$ et ROBUSTMADS$_{(\sigma^2=0,1)}$ pour des valeurs de $\alpha = 40\%$, on remarque qu'ils réagissent à l'augmentation du rayon r. Ils résolvent plus de problèmes dans un voisinage plus large. En effet, dans le graphique (4.2a), ROBUST-MADS$_{(\sigma^2=0,2)}$ résout 30% des problèmes et ROBUSTMADS$_{(\sigma^2=0,1)}$ en résout 50%. Par contre, à la sous-figure (4.2b) les deux résolvent 40% et 60% respectivement. Pour les sous-figures (4.2c) et (4.2d) le niveau du bruit est augmenté à $\sigma^2_{bruit} = 0,2$. Dans l'ensemble, les algorithmes résolvent moins de problèmes que pour un bruit plus faible. Pour un rayon proche de la solution ROBUSTMADS$_{(\sigma^2=0,01)}$ est clairement le meilleur. Il domine tous les autres algorithmes en résolvant 50% des problèmes à $\alpha = 20\%$, tandis que les autres résolvent moins de 40% des problèmes. L'algorithme MADS est le moins performant. Dans un voisinage plus large, ROBUSTMADS$_{(\sigma^2=0,1)}$ et ROBUSTMADS$_{(\sigma^2=0,01)}$ se rejoignent en résolvant 40% des problèmes à $\alpha = 20\%$. Par contre, seulement l'algorithme ROBUSTMADS$_{(\sigma^2=0,1)}$ résout plus de problèmes dans un rayon plus large.

Finalement, les graphiques (4.2e) et (4.2f) montrent le comportement des algorithmes dans le cas où le niveau du bruit est le plus élevé soit $\sigma^2_{bruit} = 1$. Il est clair que l'algorithme ROBUSTMADS$_{(\sigma^2=0,2)}$ est meilleur. Il domine les autres algorithmes et résout 50% des problèmes. Quand le rayon devient plus large le nombre de problèmes résolus diminue. On remarque aussi que l'ordre de robustesse des algorithmes est inversé par rapport aux graphiques précédents. En effet, quand le niveau du bruit passe de 0,2 à 1 l'algorithme ROBUST-MADS$_{(\sigma^2=0,01)}$ n'est pas capable de retourner une bonne solution et le lissage effectué avec

le paramètre $\sigma^2 = 0,01$ n'a pas beaucoup d'influence sur le bruit, car dans la fonction du lissage à l'équation (3.6) les points qui sont plus loin du point considéré n'influence pas la valeur lissée de la fonction étant donné que leur poids sont très proche de zéro. Mais MADS demeure celui qui retourne la solution la moins robuste.

La mesure basée sur la variance

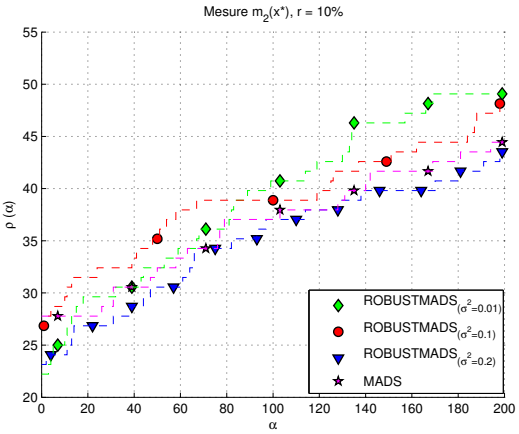
La figure (4.3) montre le comportement des algorithmes en considérant la mesure basée sur la variance pour les trois niveaux de bruit. On remarque au graphique (4.3a) que pour $\alpha < 22\%$ l'algorithme MADS résout près de 27% des problèmes et ROBUSTMADS$_{(\sigma^2=0,1)}$ en résout moins à $\alpha = 0$. Mais dès que $\alpha > 5\%$ il résout plus de 32% des problèmes. Dans le cas où le rayon $r = 10\%$, c'est ROBUSTMADS$_{(\sigma^2=0,1)}$ qui est le meilleur. Aussitôt que le rayon est augmenté, au graphique 4.3b c'est l'algorithme ROBUSTMADS$_{(\sigma^2=0,2)}$ qui est meilleur en résolvant jusqu'à 39% des problèmes pour $\alpha = 50\%$. Après cette valeur les deux algorithmes ROBUSTMADS$_{(\sigma^2=0,2)}$ et MADS ont le même comportement. Par ailleurs, l'algorithme MADS rattrape l'algorithme ROBUSTMADS$_{(\sigma^2=0,1)}$ à $\alpha = 40\%$ puis le dépasse complètement pour résoudre plus de problèmes. Par contre ROBUSTMADS$_{(\sigma^2=0,01)}$ est le moins performant.

Quand le niveau du bruit est plus élevé, le graphique (4.3c) montre que les deux algorithmes ROBUSTMADS$_{(\sigma^2=0,01)}$ et ROBUSTMADS$_{(\sigma^2=0,2)}$ se démarquent des autres. Le premier résout 37% des problèmes et le deuxième en résout 32% à $\alpha = 20\%$. Quant au autres algorithmes, ils résolvent seulement 15% des problèmes à $\alpha = 20\%$. Pour un rayon plus large, au graphique (4.3d), on remarque que pour $\alpha < 20\%$ l'algorithme ROBUSTMADS$_{(\sigma^2=0,1)}$ vient rejoindre les autres en résolvant environ 30% des problèmes. C'est ROBUSTMADS$_{(\sigma^2=0,2)}$ qui est le plus performant, il résout jusqu'à 38% des problèmes à $\alpha = 40\%$, et MADS reste le moins performant.

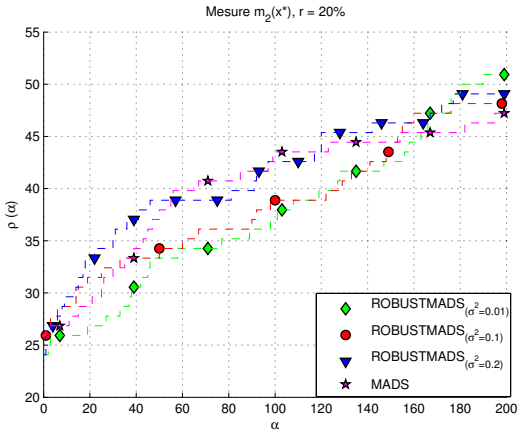
Aux graphiques (4.3e) et (4.3f) le niveau de bruit est le plus élevé et il n'y a qu'un seul algorithme qui se démarque des autres. L'algorithme ROBUSTMADS$_{(\sigma^2=0,2)}$ résout 50% des problèmes quand $\alpha < 60\%$ au (4.3e). Pour un rayon plus large c'est MADS qui prends le dessus, parmi ROBUSTMADS$_{(\sigma^2=0,01)}$ et ROBUSTMADS$_{(\sigma^2=0,1)}$, mais il demeure très mauvais devant ROBUSTMADS$_{(\sigma^2=0,2)}$ qui résout plus de problèmes que quand le rayon était moins élevé.

La mesure basée sur l'écart type

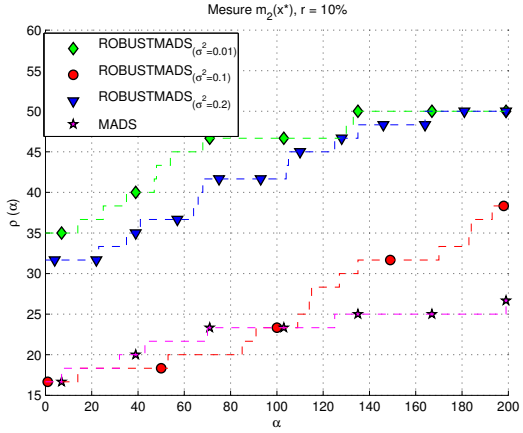
Les résultats illustrés par la figure (4.4) repose sur la mesure de l'écart type. Les graphiques (4.4a) et (4.4b) sont donnés pour un niveau de bruit faible $\sigma^2_{bruit} = 0,01$ et ils montrent que les algorithmes ROBUSTMADS$_{(\sigma^2=0,01)}$ et ROBUSTMADS$_{(\sigma^2=0,1)}$ ont un compor-



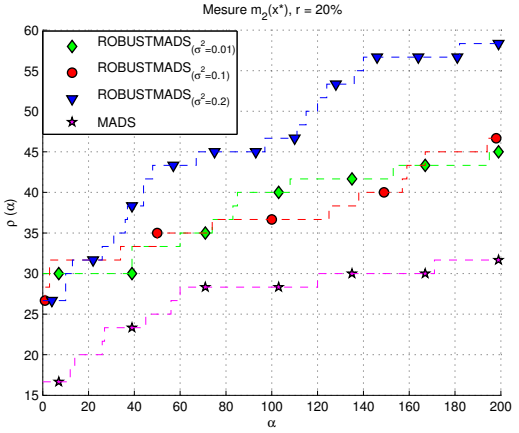
(a) $\sigma^2_{bruit} = 0,01$



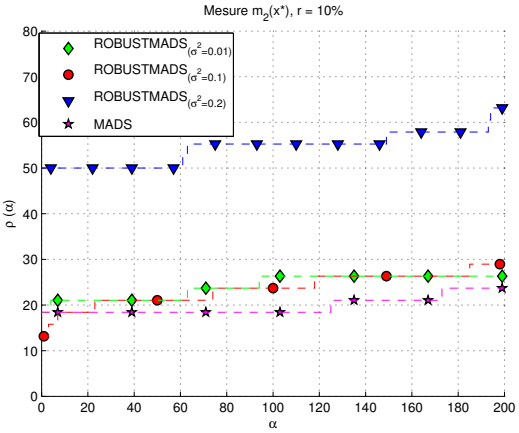
(b) $\sigma^2_{bruit} = 0,01$



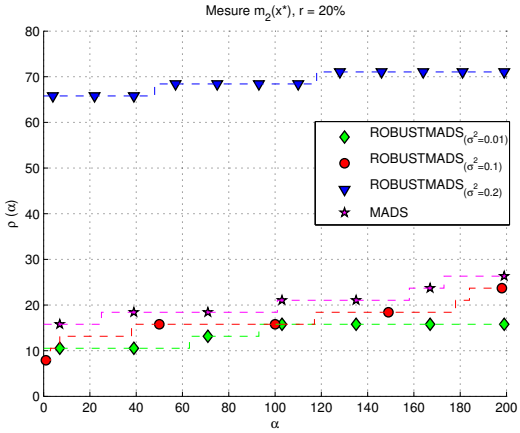
(c) $\sigma^2_{bruit} = 0,2$



(d) $\sigma^2_{bruit} = 0,2$



(e) $\sigma^2_{bruit} = 1$



(f) $\sigma^2_{bruit} = 1$

Figure 4.3 Profils de robustesse pour la mesure basée sur la variance. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\sigma^2_{bruit} \in \{0,01; 0,2; 1\}$, et la tolérance α exprimée en pourcentage.

tement similaire pour les deux rayons. Ils dominent les autres algorithmes en résolvant près de 40% des problèmes à $\alpha = 20\%$. En revanche, quand $r = 20\%$ c'est RobustMads$_{(\sigma^2=0,1)}$ qui est le plus performant.

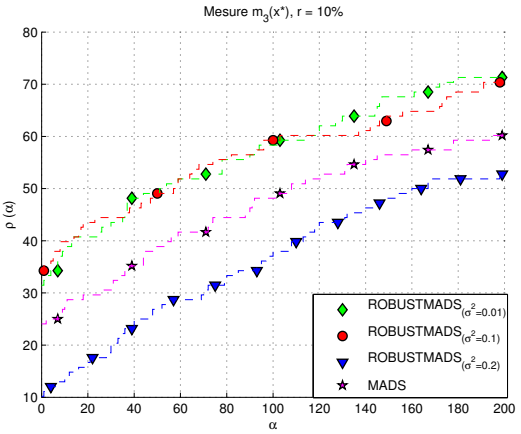
Dès que le niveau du bruit passe à 0,2, les graphiques (4.4c) et (4.4d) montrent que RobustMads$_{(\sigma^2=0,01)}$ résout plus de problèmes que quand le bruit était plus faible, soit 47% des problèmes à $\alpha = 40\%$ et pour les deux rayons.

Les algorithmes sont maintenant comparés pour un niveau de bruit plus élevé, soit $\sigma^2_{bruit} = 1$. Les résultats sont donnés aux graphiques (4.4e) et (4.4e). On remarque que RobustMads$_{(\sigma^2=0,2)}$ est celui qui résout le plus de problèmes. Si on se situe à $\alpha = 30\%$, dans (4.4e) et (4.4e), il résout 37% et 35% des problèmes respectivement. Il est suivi par RobustMads$_{(\sigma^2=0,1)}$. Pour un niveau élevé de bruit Mads demeure le moins performant.

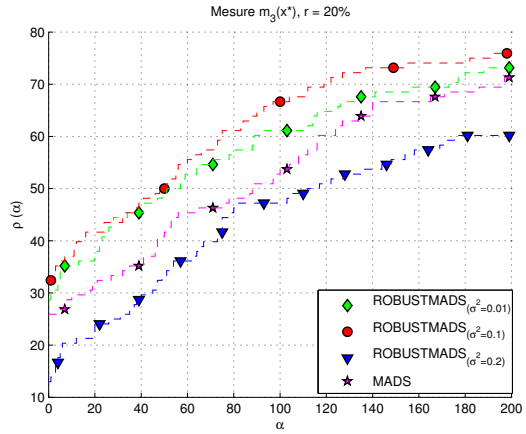
La mesure basée sur la moyenne

Les résultats présentés à la figure (4.5) on été obtenus en utilisant la dernière mesure qui est basée sur la moyenne. Pour les sous-figures (4.5a) et (4.5b), l'algorithme Mads domine le reste des algorithmes et ce pour les deux rayons $r = 10\%$ et $r = 20\%$, mais demeure très proche de l'algorithme RobustMads$_{(\sigma^2=0,01)}$ pour un niveau de bruit faible ($\sigma^2_{bruit} = 0,01$). Pour les sous figures 4.5c et 4.5d le bruit passe au niveau moyen ($\sigma^2_{bruit} = 0,2$) et on remarque que l'algorithme RobustMads$_{(\sigma^2=0,01)}$ se démarque des autres et résout près de 50% des problèmes à $\alpha = 20\%$, et entre une tolérance de $5\% \leq \alpha \leq 65\%$, Mads est dominé par tous les algorithmes. Finalement, quand le bruit passe à un niveau élevé ($\sigma^2_{bruit} = 1$) on peut voir aux sous figures (4.5e) et (4.5f) que l'algorithme RobustMads$_{(\sigma^2=0.2)}$ domine le tous les algorithmes, quant à Mads, c'est le moins performant. Nous faisons le même constat qui a été fait par l'analyse des mesures précédentes pour un bruit élevé.

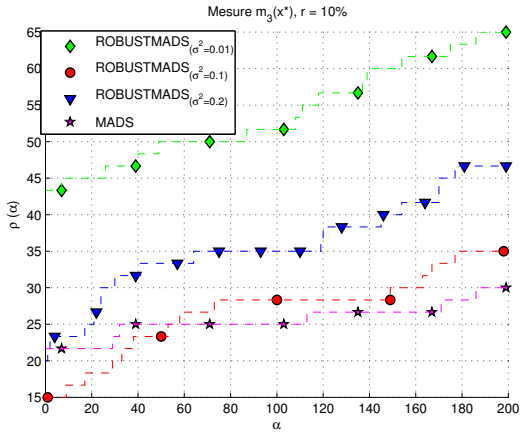
On constate que lorsque le bruit atteint un niveau élevé, dans le cas de ce travail $\sigma^2_{bruit} = 1$, c'est l'algorithme appliqué avec le paramètre de lissage le plus élevé, soit $\sigma^2 = 0,2$, qui est le plus performant. Cette remarque est la même pour chaque mesure utilisée. À la prochaine sous-section nous allons analyser le comportement des algorithmes avec un bruit de type déterministe afin de comparer les résultats aux précédents et de pouvoir tirer des conclusions.



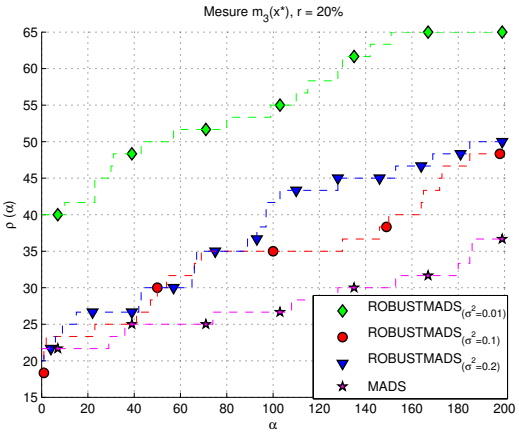
(a) $\sigma^2_{bruit} = 0,01$



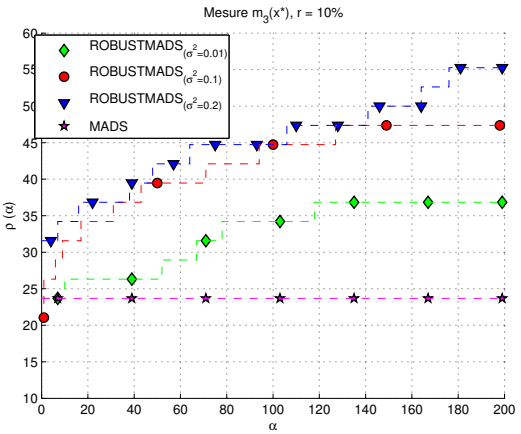
(b) $\sigma^2_{bruit} = 0,01$



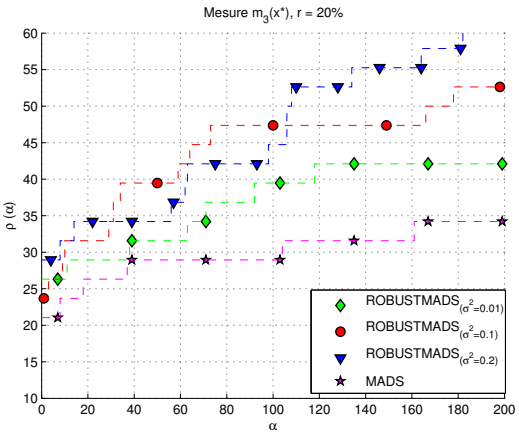
(c) $\sigma^2_{bruit} = 0,2$



(d) $\sigma^2_{bruit} = 0,2$

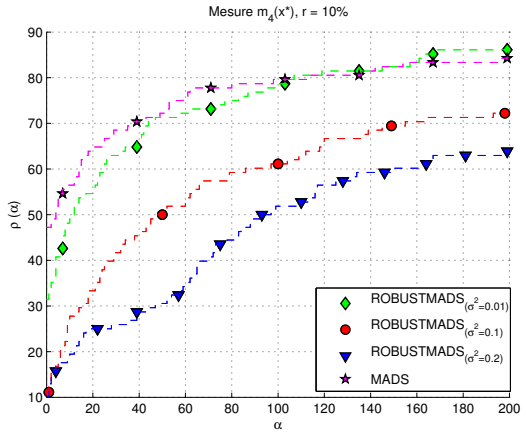


(e) $\sigma^2_{bruit} = 1$

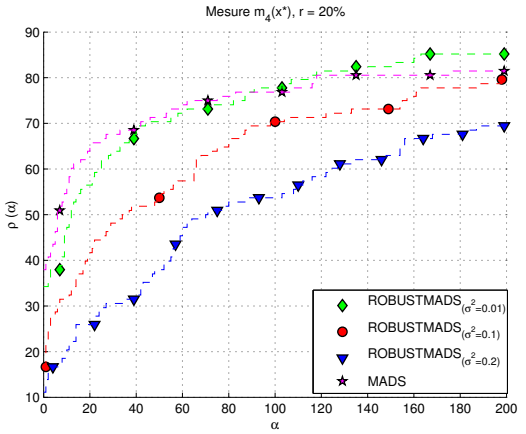


(f) $\sigma^2_{bruit} = 1$

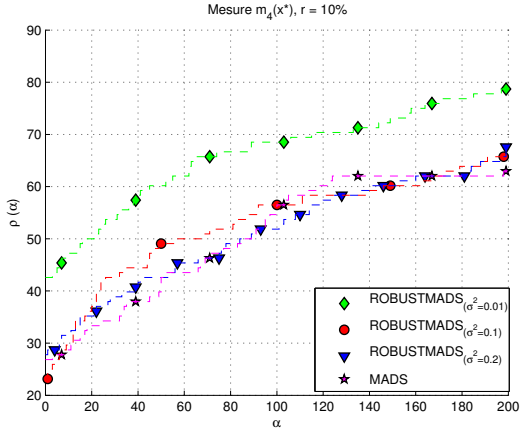
Figure 4.4 Profils de robustesse pour la mesure basée sur l'écart type. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\sigma^2_{bruit} \in \{0,01; 0,2; 1\}$ et la tolérance α exprimée en pourcentage.



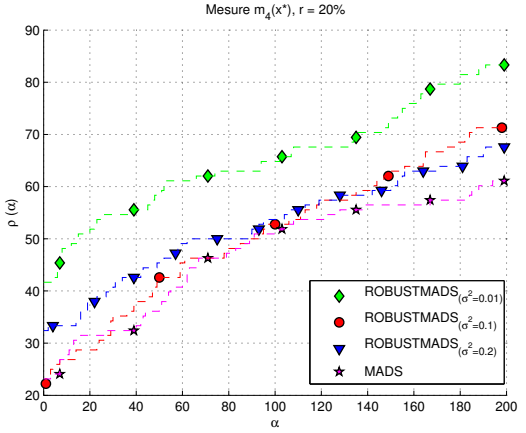
(a) $\sigma^2_{bruit} = 0,01$



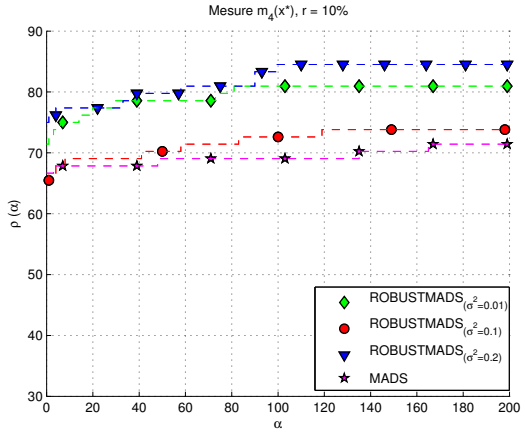
(b) $\sigma^2_{bruit} = 0,01$



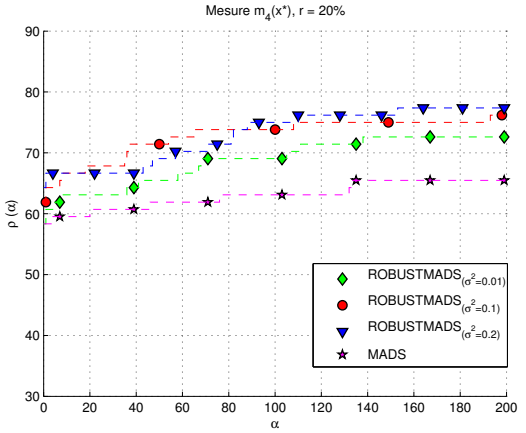
(c) $\sigma^2_{bruit} = 0,2$



(d) $\sigma^2_{bruit} = 0,2$



(e) $\sigma^2_{bruit} = 1$



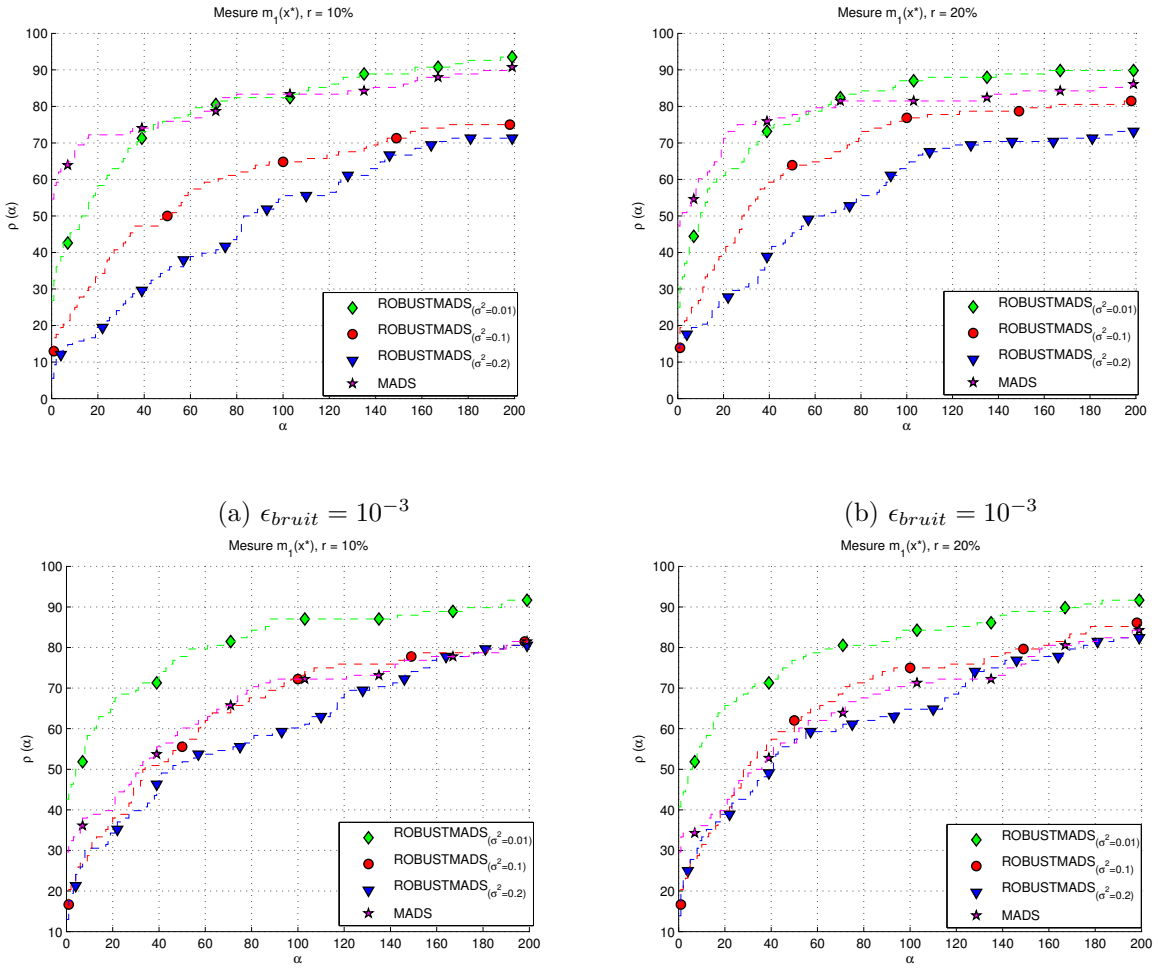
(f) $\sigma^2_{bruit} = 1$

Figure 4.5 Profils de robustesse pour la mesure basée sur la moyenne. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\sigma^2_{bruit} \in \{0,01; 0,2; 1\}$ et la tolérance α exprimée en pourcentage.

4.3.2 Résultats pour les problèmes déterministes

Les résultats donnés ici sont une analyse des solutions en considérant les problèmes déterministe (4.5), et en utilisant les mêmes mesures de robustesse que pour les problèmes stochastiques et en faisant varier le niveau de bruit soit $\epsilon_{bruit} \in \{10^{-1},\ 10^{-3}\}$.

La mesure du pire cas



(a) $\epsilon_{bruit} = 10^{-3}$ (b) $\epsilon_{bruit} = 10^{-3}$

(c) $\epsilon_{bruit} = 10^{-1}$ (d) $\epsilon_{bruit} = 10^{-1}$

Figure 4.6 Profils de robustesse pour la mesure du pire cas. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\epsilon_{bruit} \in \{10^{-3},\ 10^{-1}\}$ et la tolérance α exprimée en pourcentage.

La figure (4.6) regroupe les résultats d'un problème déterministe lorsque la mesure du pire cas est utilisée. Le graphique 4.6a montre que quand le rayon et le bruit sont faibles, MADS résout environ 70% des problèmes, à $\alpha < 40\%$, et il est rattrapé par ROBUSTMADS$_{(\sigma^2=0,01)}$ à

$\alpha = 40\%$. Les deux autres algorithmes sont moins performant. Lorsque le rayon du voisinage de la solution augmente, à la figure (4.6b), l'algorithme RobustMads$_{(\sigma^2=0,01)}$ se rapproche plus de Mads en résolvant 57% des problems alors que Mads en résout 60% pour $\alpha = 10\%$. Les deux autres demeurent moins performants, néanmoins ils résolvent globalement plus de problèmes.

À la figure (4.6c) et (4.6d) le niveau du bruit est élevé à 10^{-1}, on remarque que le meilleur algorithme est RobustMads$_{(\sigma^2=0,01)}$, il résout près de 70% des problèmes à $\alpha = 20\%$ et il n'est pas affecté par l'augmentation du rayon, en effet, les deux courbes à la figure (4.6c) et (4.6d) sont très similaires. Mads prend la deuxième position en résolvant 40% des problèmes à $\alpha = 20\%$ comme le montre la figure (4.6c), mais quand le rayon augmente à la figure (4.6d), RobustMads$_{(\sigma^2=0,1)}$ dépasse Mads à $\alpha = 21\%$.

La mesure basée sur la variance

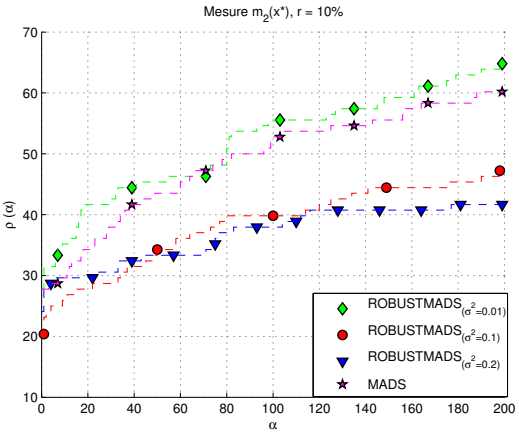
Quand le bruit est faible on remarque aux deux figures (4.7a) et (4.7b) que l'algorithme RobustMads$_{(\sigma^2=0,01)}$ est toujours proche de Mads. Mais la mesure de l'écart type montre qu'avec RobustMads$_{(\sigma^2=0,01)}$ plus de problèmes retournent des solutions robustes qu'avec Mads. Les autres algorithmes ont le même comportement dans un rayon élevé pour des valeurs de $\alpha < 60\%$. Les deux autres figures (4.7c) et (4.7d), illustrent les résultats lorsque le niveau du bruit est augmenté. On remarque que RobustMads$_{(\sigma^2=0,01)}$ se démarque des autres algorithmes en résolvant, à $\alpha = 40\%$, environ 55% et 50% des problèmes pour les rayons 10% et 20% respectivement.

La mesure basée sur l'écart type

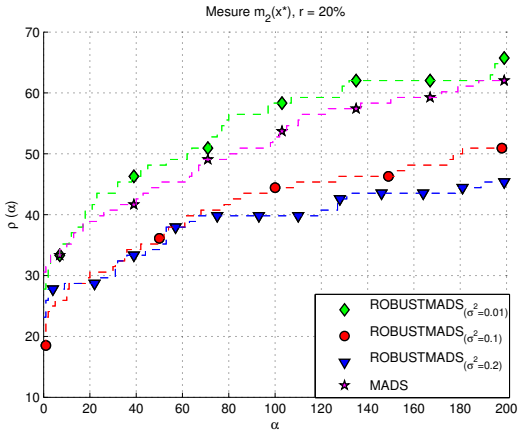
Les résultats obtenues en utilisant la mesure de l'écart type sont donnés par la figure 4.8. Quand le bruit est faible les algorithmes sont très proches, néanmoins, c'est l'algorithme RobustMads$_{(\sigma^2=0,01)}$ qui offre plus de problèmes ayant des solutions robustes. Et dès que le niveau du bruit est augmenté il se démarque des autres et ce pour les deux rayons. En se donnant une tolérance $\alpha = 40\%$, il résout environ 65% et près de 70% des problèmes pour un rayon de 10% et 20% respectivement.

La mesure basée sur la moyenne

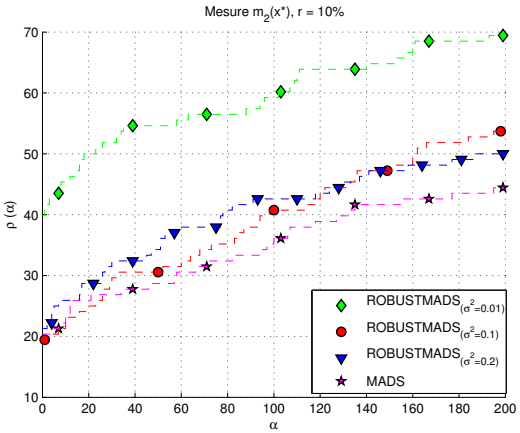
À la figure (4.9) sont illustrés les résultats obtenus en utilisant la mesure basée sur la moyenne. On remarque que lorsque le bruit est déterministe et faible ($\epsilon_{bruit} = 10^{-3}$), il n'est pas primordial d'effectuer le lissage, en effet, Mads domine tous les algorithmes pour les deux



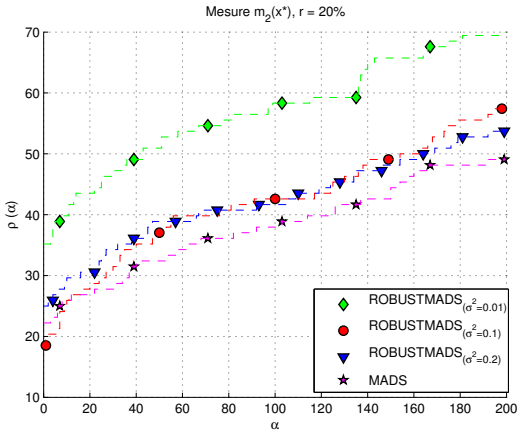
(a) $\epsilon_{bruit} = 10^{-3}$



(b) $\epsilon_{bruit} = 10^{-3}$



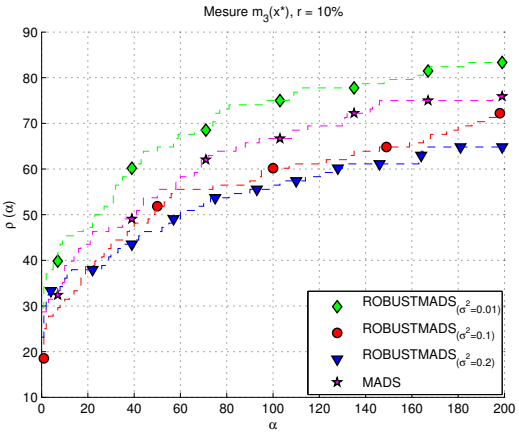
(c) $\epsilon_{bruit} = 10^{-1}$



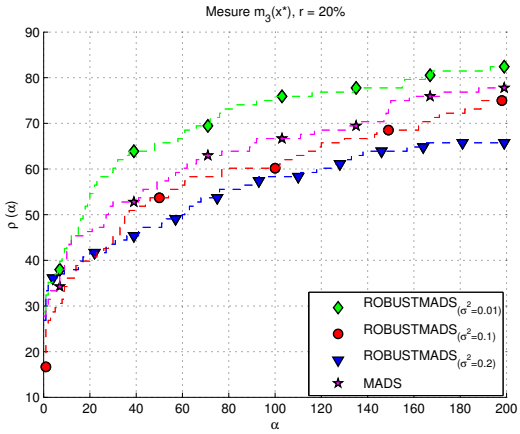
(d) $\epsilon_{bruit} = 10^{-1}$

Figure 4.7 Profils de robustesse pour la mesure basée sur la variance. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\epsilon_{bruit} \in \{10^{-3},\ 10^{-1}\}$ et la tolérance α exprimée en pourcentage.

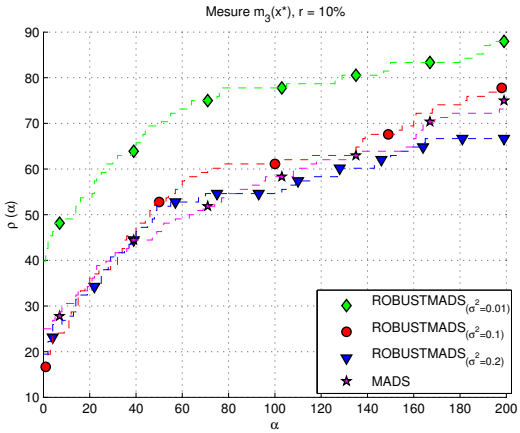
rayons et pour des valeurs de $\alpha \leq 60\%$. Lorsque le bruit devient plus fort ($\epsilon_{bruit} = 10^{-1}$), cette mesure montre que c'est l'algorithme RobustMads$_{(\sigma^2=0,01)}$ qui résout le plus de problèmes, soit 75% des problèmes pour une tolérance de $\alpha = 40\%$. Même dans le cas déterministe l'analyse des résultats de cette mesure rejoint les analyses des résultats des mesures qui précèdent.



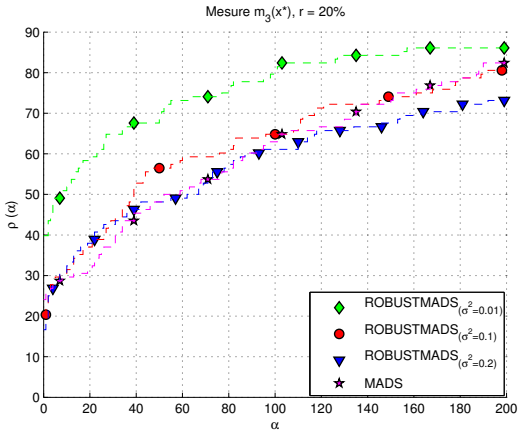
(a) $\epsilon_{bruit} = 10^{-3}$



(b) $\epsilon_{bruit} = 10^{-3}$



(c) $\epsilon_{bruit} = 10^{-1}$

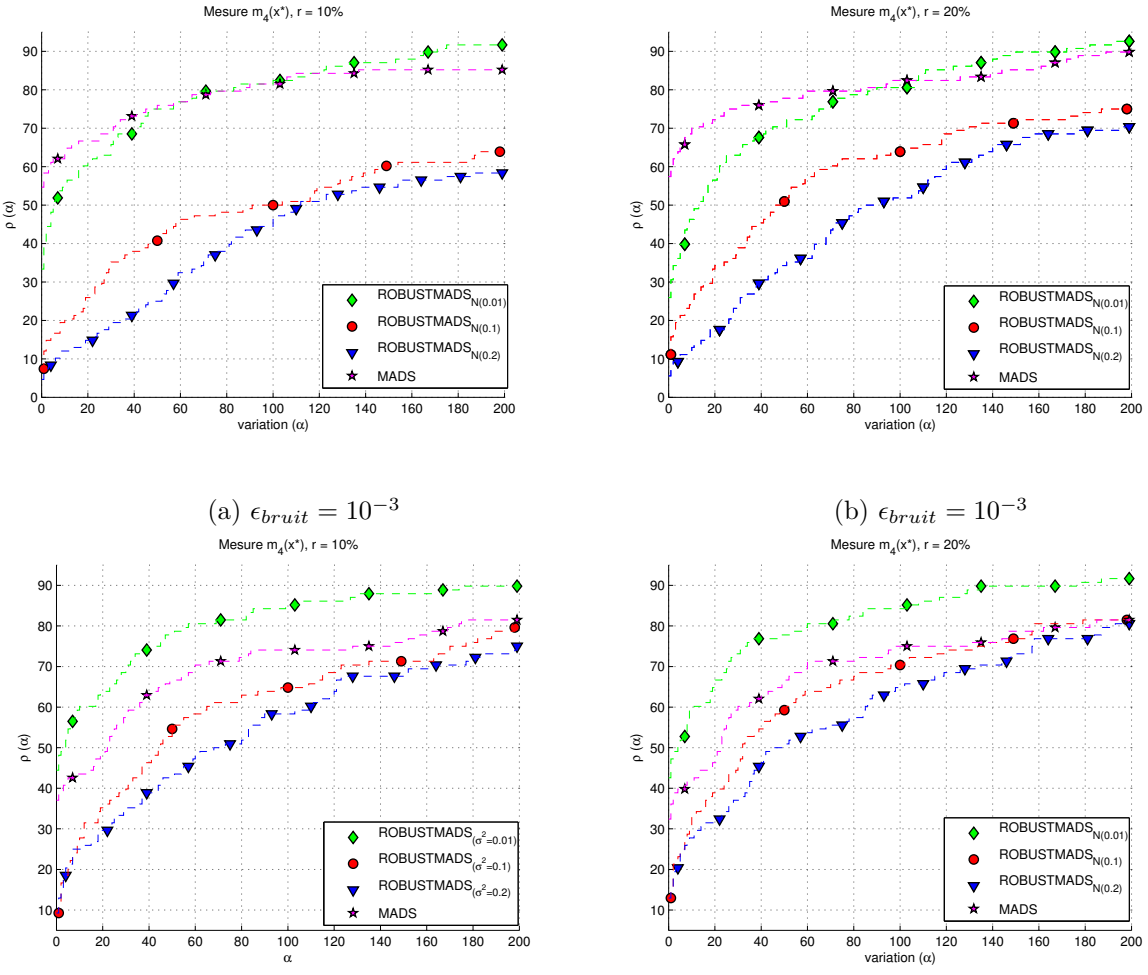


(d) $\epsilon_{bruit} = 10^{-1}$

Figure 4.8 Profils de robustesse pour la mesure basée sur l'écart type. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\epsilon_{bruit} \in \{10^{-3},\ 10^{-1}\}$ et la tolérance α exprimée en pourcentage.

4.4 Synthèse des résultats

Dans un domaine incertain la robustesse permet de donner une garantie par rapport à la solution obtenue. Les deux mesures naturelles de la robustesse qu'on peut définir par rapport à la solution qu'on vise à obtenir, sont la mesure de l'écart type et la mesure de la moyenne. L'une permet de quantifier la dispersion autour de la valeur de l'objectif et l'autre la valeur moyenne de la fonction autour d'une solution donnée. Cette section présente une synthèse des résultats obtenus en utilisant deux autres mesures qui s'ajoutent à celles qu'on vient de citer, soit la mesure du pire cas et la mesure basée sur la variance. L'analyse des résultats montre



(a) $\epsilon_{bruit} = 10^{-3}$

(b) $\epsilon_{bruit} = 10^{-3}$

(c) $\epsilon_{bruit} = 10^{-1}$

(d) $\epsilon_{bruit} = 10^{-1}$

Figure 4.9 Profils de robustesse pour la mesure basée sur la moyenne. Le rayon $r \in \{10\%, 20\%\}$, le niveau de bruit $\epsilon_{bruit} \in \{10^{-3},\ 10^{-1}\}$ et la tolérance α exprimée en pourcentage.

que l'algorithme RobustMads n'a pas un comportement totalement semblable quant au type de bruit.

Premièrement, nous avons mené une série de tests avec un bruit de type stochastique et les résultats que nous avons obtenus, par quatre mesures de robustesse différentes, se rejoignent tous pour une même conclusion. Quand le niveau du bruit est élevé le paramètre de lissage le plus élevé est celui qui permet à l'algorithme de retourner une solution robuste. En effet, le paramètre de lissage interprète le niveau d'aplatissement de la courbe de la loi normale, qui est utilisée dans la formule du lissage (3.6) et pour le paramètre $\sigma^2 = 0,2$ la courbe est légèrement plus aplatie comparé aux autres courbes dont les paramètres sont : $\sigma^2 = 0,01$ et

$0,1$. C'est-à-dire que plus de points, dans l'équation (3.6) de la fonction lissée, auront une contribution pour le calcul de la valeur lissée car leur poids ne sont pas très proches de zéro.

Par contre, si le bruit est faible, le fait de choisir un paramètre élevé pour le lissage ne donne pas de bons résultats, en effet, l'algorithme RobustMads$_{(\sigma^2=0,2)}$ est classé parmi les moins performants pour toutes les mesures à un niveau de bruit faible et moyen. Par ailleurs, en se basant sur la mesure du pire cas et la mesure basée sur la moyenne, en considérant le cas où le niveau du bruit est faible, l'algorithme RobustMads$_{(\sigma^2=0,01)}$ est comparable à Mads. Pour la mesure basée sur la variance les algorithmes ne se distinguent pas vraiment donc le lissage n'a pas d'utilité lorsque le bruit est très faible. Par contre, si on s'intéresse aux résultats donnés par la mesure de l'écart type, quoiqu'ils sont similaires aux résultats donnés par les autres mesures pour un des niveaux de bruit moyen et fort, mais pour un bruit faible ils montrent que RobustMads$_{(\sigma^2=0,01)}$ et RobustMads$_{(\sigma^2=0,1)}$ sont meilleurs. Il est certain que la solution obtenue en effectuant un lissage avec les paramètres $\{0,01; 0,1\}$ est plus stable, mais moins performante, c'est ce que révèle la mesure du pire cas. En effet, on cherche une solution qui est à la fois stable et qui donne une bonne performance donc, ce fier uniquement à une seule mesure peut être trompeur.

Deuxièmement, l'algorithme est testé pour des problèmes déterministes et les résultats montrent qu'il est recommandé d'effectuer un lissage avec le paramètre $\sigma^2 = 0,01$ quand le bruit est élevé, et comme pour les résultats obtenus quand le bruit est stochastique, il n'est pas recommandé d'appliquer le processus de lissage quand le bruit est faible. Par contre RobustMads$_{(\sigma^2=0,2)}$ n'est pas le plus performant lorsque le bruit est élevé.

On en déduit que l'approche proposée dans ce travail est capable de trouver une solution robuste pour tout type de bruit et notamment lorsque le bruit est de type stochastique et de niveau élevé. Ainsi, les résultats analysés dans ce chapitre confirment l'intérêt pratique du lissage.

CHAPITRE 5

CONCLUSION

Ce chapitre se veut une synthèse du travail effectué dans le cadre de la maîtrise en exposant la contribution principale, et pour finir, des ouvertures et des améliorations futures sont proposées.

On a traité les problèmes d'optimisation sans contraintes et sous incertitude où les fonctions sont vues comme des boîtes-noires. Cette problématique est souvent présente dans les problèmes industriels. Le travail présenté ici montre une façon de limiter l'influence du bruit qui affecte les solutions, quand nous avons seulement accès aux valeurs de la fonction objectif. D'abord, on a proposé un survol des méthodes d'optimisation sans dérivées qui représentent le coeur des méthodes utilisées. Puis, on a présenté une brève introduction sur les méthodes d'optimisation sous incertitude, dont les différentes formulations d'un problème d'optimisation robuste et ce, après avoir détaillé les méthodes directes directionnelles comme l'algorithme Mads sur lequel repose notre principale méthode.

Par la suite, le chapitre 3 est consacré aux détails de l'algorithme proposé qui consiste à effectuer une opération de lissage qui intervient à chaque itération de l'algorithme en faisant une somme pondérée des anciennes évaluations. Cette opération est judicieuse lorsque les fonctions sont coûteuses en temps de calcul et le budget est très limité en nombre d'évaluations. Aussi, dans ce chapitre, une brève analyse de convergence de l'algorithme basée sur les résultats fondamentaux de la convergence de Mads a été donnée.

Cette approche peut trouver son analogue dans la littérature, où de nombreuses méthodes conçues spécifiquement pour palier au problème de l'incertitude sont proposées mais requièrent la connaissance de l'expression analytique de la fonction à optimiser.

L'intérêt d'utiliser une technique de lissage, dans l'optimisation de fonctions bruitées, est démontré au chapitre 4, ainsi que l'efficacité de la méthode lorsque le niveau du bruit est assez élevé.

Dans un contexte incertain, il est souhaitable d'offrir à l'utilisateur une certaine garantie sur la solution qu'il désire obtenir. Ce travail propose quatre façons de quantifier la robustesse d'une solution à un problème où les évaluations sont souvent erronées à cause du bruit qui affecte la fonction objectif. Cette quantification de la robustesse nous permet non seulement de comparer les solutions obtenues, mais aussi de fixer le paramètre du lissage adéquat au type et au niveau du bruit afin de pouvoir donner des recommandations.

5.1 Contribution

Au cours de ce travail de maîtrise, l'objectif principal était de mettre en place une technique permettant à la fois de donner une solution performante et robuste pour les problèmes d'optimisation de boîte-noire bruitée. On a proposé une technique de lissage, qui constitue une contribution importante de ce travail, et qui s'est avérée assez prometteuse lorsque le bruit est à un niveau élevé et à caractère stochastique. De plus, étant donné que nous disposons d'un budget d'évaluations restreint, cette technique ne requière aucune évaluation supplémentaire.

Ce procédé de lissage, qui opère au cours du processus d'optimisation, a fait l'objet du développement d'un nouvel algorithme de recherche directe robuste soit l'algorithme RobustMads implémenté en C++ en utilisant le logiciel NOMAD.

Le constat effectué au chapitre 4 mesure la performance de l'algorithme proposé et, par conséquent, la qualité de la solution. Effectivement, il montre qu'une perturbation des variables d'entrées ne conduit pas à une grande dégradation des résultats de l'optimisation. Ce constat a été confirmé pour les différentes mesures utilisées.

5.2 Ouvertures de recherche

Les résultats des travaux présentés dans ce mémoire ouvrent la porte à plusieurs approches possibles pour la gestion du bruit dans l'optimisation de fonctions coûteuses. Ils constituent un point de départ pour la mise en place d'un système permettant la gestion du bruit en le réduisant ou en le filtrant selon son intensité et son type. La mise en œuvre d'un tel système nécessite d'apporter des réponses à de nombreuses interrogations et des connaissances du modèle à optimiser. Aussi, la mesure de robustesse peut être intégrée directement dans le modèle.

Il est démontré, à partir des résultats de l'approche proposée ici, que lorsque le bruit est faible, l'application du lissage apporte peu. Aussi, étant donné que l'intensité du bruit n'est souvent pas la même pour tous les points et pour chaque évaluation, une perspective intéressante se dégage. Il serait intéressant d'intégrer l'intensité du bruit comme information supplémentaire dans le processus de lissage. L'obtention de cette information nécessite d'avoir recours à des méthodes statistiques pour l'anticiper. De cette façon, le lissage du bruit se fera de façon dynamique.

Par ailleurs, il importe aussi d'explorer l'effet de l'approche proposée sur d'autres types de problèmes. La solution proposée a été testée sur des problèmes sans contraintes. Il serait intéressant d'étendre le travail aux problèmes avec contraintes de bornes, puis aux problèmes avec contraintes en considérant la présence du bruit dans les contraintes qui ont aussi le

comportement de boîtes-noires.

La solution proposée ici est donnée pour des problèmes académiques et n'est pas encore applicable sur un problème industriel. En effet, en industrie, il est plutôt rare de traiter des problèmes mono-objectif sans contraintes et souvent on ne possède d'aucune information sur le bruit. Il convient alors d'explorer aussi cette piste. Les solutions robustes sont très recherchées en industrie, il est nécessaire pour beaucoup d'entreprises de proposer des solutions qui répondent à leurs attentes en matière de robustesse. On est intéressé par l'élaboration d'algorithme de boîtes-noires fournissant de telles solutions.

RÉFÉRENCES

ABRAMSON, M. (2004). Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. *Optimization and Engineering*, 5, 157–177.

ABRAMSON, M., AUDET, C., COUTURE, G., DENNIS, JR., J., LE DIGABEL, S. et TRIBES, C. (2014). The NOMAD project. Software available at `https://www.gerad.ca/nomad`.

ABRAMSON, M., AUDET, C. et DENNIS, JR., J. (2004). Generalized pattern searches with derivative information. *Mathematical Programming*, Series B, 100, 3–25.

ALARIE, S., AUDET, C., GARNIER, V., LE DIGABEL, S. et LECLAIRE, L.-A. (2013). Snow water equivalent estimation using blackbox optimization. *Pacific Journal of Optimization*, 9, 1–21.

ANDERSON, E. J. et FERRIS, M. C. (2001). A direct search algorithm for optimisation with noisy function evaluations. *Society for Industrial and Applied Mathematics*, vol. 11, pp. 837–857.

ARMBRUSTER, B. et DELAGE, E. (2014). Decision making under uncertainty when preference information is incomplete. *Management Science (à paraître)*.

AUDET, C. (2011). A short proof on the cardinality of maximal positive bases. *Optimization Letters*, 5, 191–194.

AUDET, C. (2014). A survey on direct search methods for blackbox optimization and their applications. P. M. Pardalos et T. M. Rassias, éditeurs, *Mathematics Without Boundaries*, Springer New York. 31–56.

AUDET, C., BÉCHARD, V. et LE DIGABEL, S. (2008a). Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41, 299–318.

AUDET, C., DANG, C.-K. et ORBAN, D. (2010). Algorithmic parameter optimization of the DFO method with the OPAL framework. *Software Automatic Tuning : From Concepts to State-of-the-Art Results*, Springer, chapitre 15. K. Naono and k. Teranishi and j. Cavazos and r. Suda édition, 255–274.

AUDET, C. et DENNIS, J. (2001). Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11, 573–594.

AUDET, C. et DENNIS, J. (2003). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13, 889–903.

AUDET, C. et DENNIS, J. (2006). Mesh adaptative direct search algorithms for constrained optimisation. *Society for Industrial and Applied Mathematics*, vol. 17, pp. 188–217.

AUDET, C., DENNIS, J. et LE DIGABEL, S. (2008b). Parallel space decomposition of the mesh adaptive direct search algorith. *SIAM Journal on Optimization*, 19, 1150–1170.

AUDET, C. et DENNIS, JR., J. (2004). A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14, 980–1010.

BEN-TAL, A. et NEMIROVSKI, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88, 411–424.

BEN-TAL, A. et NEMIROVSKI, A. (2002). Robust optimization - methodology and applications. *Mathematical Programming*, 92, 453–480.

BEYER, H.-G. et SENDHOFF, B. (2007). Robust optimization-a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196, 3190 – 3218.

BRIGGS, W. L., HENSON, V. E. et MCCORMICK, S. F. (2000). *A Multigrid Tutorial, Second Edition*. Society for Industrial and Applied Mathematics, seconde édition.

CHICK, S. E., INOUE, K., INOUE, K. et INOUE, K. (2001). New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49, 732–743.

CICCAZZO, A., LATORRE, V., LIUZZI, G., LUCIDI, S. et RINALDI, F. (2013). Derivative-free robust optimization for circuit design. *Journal of Optimization Theory and Applications*, 1–20.

CLARKE, F. (1983). *Optimization and Nonsmooth Analysis*. Wiley, New York. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

CONN, A. R., SCHEINBERG, K. et VICENTE, L. N. (2009). *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics.

DAVIS, C. (1954). Theory of positive linear dependence. *American Journal of Mathematics*, 76, 733–746.

DENG, G. et FERRIS, M. (2007). Extension of the direct optimization algorithm for noisy functions. *Simulation Conference, 2007 Winter*. 497–504.

DENG, G. et FERRIS, M. C. (2006). Adaptation of the uobyqa algorithm for noisy functions. *Proceedings of the 38th Conference on Winter Simulation*. Winter Simulation Conference, WSC '06, 312–319.

ELSTER, C. et NEUMAIER, A. (1995). A grid algorithm for bound constrained optimization of noisy functions. *IMA Journal of Numerical Analysis*, 15, 585–608.

FERMI, E. et METROPOLIS, N. (1952). Rapport technique, Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA.

GOERIGK, M. (2014). Ropi a robust optimization programming interface for c. *Optimization Methods and Software*, 29, 1261–1280.

HASTIE, T., TIBSHIRANI, R. et FRIEDMAN, J. H. (2001). *The elements of statistical learning : data mining, inference, and prediction*. Springer.

JONES, D., PERTTUNEN, C. et STUCKMAN, B. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79, 157–181.

KOKKOLARAS, M., AUDET, C. et DENNIS, J.E., J. (2001). Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering*, 2, 5–29.

KOSTROWICKI, J., PIELA, L., CHERAYIL, B. J. et SCHERAGA, H. A. (1991). Performance of the diffusion equation method in searches for optimum structures of clusters of lennard-jones atoms. *The Journal of Physical Chemistry*, 95, 4113–4119.

LARSON, J. et BILLUPS, S. C. (2013). Stochastic derivative-free optimisation using a trust region framework. vol. 00, pp. 95–102.

LARSON, J. et WILD, S. M. (2013). Non-intrusive termination of noisy optimization. *Optimization Methods Software*, 28, 993–1011.

LE DIGABEL, S. (2011). Algorithm 909 : NOMAD : Nonlinear optimization with the mads algorithm. *ACM Trans. Math. Softw.*, 37, 1–15.

LE DIGABEL, S., ABRAMSON, M., AUDET, C. et DENNIS, JR., J. (2010). Parallel versions of the MADS algorithm for black-box optimization. *Optimization days*. Montreal. Slides available at `www.gerad.ca/Sebastien.Le.Digabel/talks/2010_JOPT_25mins.pdf`.

LEE, K. H. et PARK, G. J. (2001). Robust optimization considering tolerances of design variables. *Computers and Structures*, 79, 77 – 86.

LEWIS, R. et TORCZON, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9, 1082–1099.

LEWIS, R. et TORCZON, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10, 917–941.

LEWIS, R. M. et TORCZON, V. (1996). Rank ordering and positive bases in pattern search algorithms. Rapport technique, DTIC Document.

LI, J., WU, C., WU, Z. et LONG, Q. (2014). Gradient-free method for nonsmooth distributed optimization. *Journal of Global Optimization*, 1–16.

LIU, Q. (2013). Linear scaling and the direct algorithm. *Journal of Global Optimization*, 56, 1233–1245.

LIU, Q., ZENG, J. et YANG, G. (2014). Mrdirect : a multilevel robust direct algorithm for global optimization problems. *Journal of Global Optimization*, 1–23.

MARAZZI, M. et NOCEDAL, J. (2002). Wedge trust region methods for derivative free optimization. *Mathematical Programming*, 91, 289–305.

MARSDEN, A. (2004). *Aerodynamic Noise Control by Optimal Shape Design*. Thèse de doctorat, Stanford University.

MARSDEN, A., WANG, M., DENNIS, JOHNE., J. et MOIN, P. (2004). Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering*, 5, 235–262.

MCKAY, M. D., BECKMAN, R. J. et CONOVER, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, 239–245.

MLADENOVIĆ, N. et HANSEN, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24, 1097–1100.

MORÉ, J. et WILD, S. M. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20, 172–191.

MORÉ, J. et WILD, S. M. (2011). Estimating computational noise. *SIAM Journal on Scientific Computing*, 33, 1292–1314.

NADARAYA, E. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9, 141–142.

NELDER, J. A. et MEAD, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7, 308–313.

POWELL, M. (2002). Uobyqa : unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92, 555–582.

POWELL, M. (2003). On trust region methods for unconstrained minimization without derivatives. *Mathematical Programming*, 97, 605–623.

POWELL, M. (2006). *The NEWUOA software for unconstrained optimization without derivatives*, vol. 83 de *Nonconvex Optimization and Its Applications*. Springer US.

POWELL, M. (2010). On the convergence of a wide range of trust region methods for unconstrained optimization. *IMA Journal of Numerical Analysis*, 30, 289–301.

POWELL, M. (2013). Beyond symmetric broyden for updating quadratic models in minimization without derivatives. *Mathematical Programming*, 138, 475–500.

RHEIN, B., CLEES, T. et RUSCHITZKA, M. (2014). Robustness measures and numerical approximation of the cumulative density function of response surfaces. *Communications in Statistics - Simulation and Computation*, 43, 1–17.

SHAO, C.-S., BYRD, R., ESKOW, E. et SCHNABEL, R. (1997). Global optimization for molecular clusters using a new smoothing approach. L. Biegler, A. Conn, T. Coleman et F. Santosa, éditeurs, *Large-Scale Optimization with Applications*, Springer New York, vol. 94 de *The IMA Volumes in Mathematics and its Applications*. 163–199.

SOYSTER, A. L. (1973). Technical note-convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21, 1154–1157.

SPILLANE, M. C., GICA, E. et TITOV, V. V. (2009). Tsunameter Network Design for the U.S. DART Array. *AGU Fall Meeting Abstracts*, A1368.

SRIVER, T. A., CHRISSIS, J. W. et ABRAMSON, M. A. (2009). Pattern search ranking and selection algorithms for mixed variable simulation-based optimization. *European Journal of Operational Research*, 198, 878 – 890.

TORCZON, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7, 1–25.

WALD, A. (1945). Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 46, pp. 265–280.

WALKER, M. et HAMILTON, R. (2005). A technique for optimally designing fibre reinforced laminated plates with manufacturing uncertainties for maximum buckling strength. *Engineering Optimization*, 37, 135–144.

WANG, M. et ZOU, X. (2010). Notice of retraction searching for robust optimal solutions by an evolutionary algorithm. *Advanced Computer Control (ICACC), 2010 2nd International Conference on*. vol. 5, 593–595.

WU, Z. (1996). The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM Journal on Optimization*, 6, 748–768.

XAVIER, V., FRANÇA, F., XAVIER, A. et LIMA, P. (2014). A hyperbolic smoothing approach to the multisource weber problem. *Journal of Global Optimization*, 60, 49–58.

YANG, W., FEINSTEIN, J. et MARSDEN, A. (2010). Constrained optimization of an idealized y-shaped baffle for the fontan surgery at rest and exercise. *Computer Methods in Applied Mechanics and Engineering*, 199, 2135–2149.